# Red Hat Enterprise Linux 6
# Guia de Ajuste de Desempenho

Otimizando rendimento de subsistema no Red Hat Enterprise Linux 6
Edição 4.0

Red Hat Peritos do Assunto em Pauta

# Red Hat Enterprise Linux 6 Guia de Ajuste de Desempenho

---

## Otimizando rendimento de subsistema no Red Hat Enterprise Linux 6
## Edição 4.0

Red Hat Peritos do Assunto em Pauta

**Editado por**

Don Domingo

Laura Bailey

**Resumo**

O Guia de Ajuste de Desempenho descreve como otimizar o desempenho de um sistema executando um Red Hat Enterprise Linux 6. Ele também documenta as atualizações relacionadas ao desempenho no Red Hat Enterprise Linux 6. Embora este guia contenha procedimentos que são testadas em campo e comprovados, a Red Hat recomenda que você teste corretamente todas as configurações programadas em um ambiente de teste antes de aplicá-lo para um ambiente de produção. Você também deve fazer backup de todos os seus dados e configurações de pré-ajuste.

# Índice

# Prefácio

## 1. Convenções de Documentos

Este manual usa diversas convenções para destacar certas palavras e frases e chamar a atenção para informações específicas.

Em PDF e edições de texto, este manual usa tipos de letras retiradas do conjunto Liberation Fonts. O conjunto de Fontes Liberation Fonts, também é usado em formato HTML, caso o conjunto esteja instalado em seu sistema. Caso ainda não esteja, como forma alternativa, estão disponíveis tipos de letras equivalentes. Nota: O Red Hat Enterprise Linux 5 e versões mais recentes do mesmo, incluem o conjunto Liberation Fonts por padrão.

### 1.1. Convenções Tipográficas

São usadas quatro convenções tipográficas para realçar palavras e frases específicas. Estas convenções, e circunstâncias a que se aplicam, são as seguintes:

**`Negrito Espaço Único`** (Mono-spaced Bold)

Usada para realçar entradas do sistema, incluindo comandos de shell, nomes de arquivos e caminhos. São também usadas para realçar teclas Maiúsculas/Minúsculas e as combinações de teclas. Por exemplo:

> Para ver o conteúdo do arquivo **`my_next_bestselling_novel`** em sua pasta de trabalho atual, insira o comando **`cat my_next_bestselling_novel`** na janela de solicitação e pressione **`Enter`** para executar o comando.

O comando acima inclui um nome de arquivo, um comando de shell e uma tecla, todos apresentados em Negrito Espaço Único (Mono-spaced Bold) e todos distintos, graças ao conteúdo.

As combinações de tecla podem ser diferenciadas de uma tecla individual pelo sinal positivo que conecta cada parte da combinação da tecla. Por exemplo:

> Pressione **`Enter`** para executar o comando.
>
> Pressione **`Ctrl`**+**`Alt`**+**`F2`** para trocar ao terminal virtual.

A primeira sentença, destaca uma tecla específica a ser pressionada. A segunda destaca duas combinações de teclas: um conjunto de três teclas pressionadas simultaneamente.

Caso o código fonte seja discutido, serão apresentados como acima, os nomes de classe, métodos, funções, nomes de variantes e valores retornados mencionados em um parágrafo, em **`Negrito de Espaço Único (Mono-spaced Bold)`**. Por exemplo:

> Classes baseadas em arquivo, incluem **`filesystem`** para sistemas de arquivo, **`file`** para arquivos, e **`dir`** para diretórios. Cada classe possui seu conjunto próprio de permissões associadas.

**Negrito Proporcional**

Esta representa as palavras e frases encontradas no sistema, incluindo os nomes de aplicativos, texto de caixa de diálogo, botões rotulados, caixa de seleção e rótulos de botão de opção, títulos de menus e sub-menus. Por exemplo:

> Escolha **Sistema** → **Preferências** → **Mouse** da barra do menu principal para lançar **Mouse Preferences**. Na aba **Botões** selecione o **Botão da esquerda do mouse** selecione a caixa e cliquem em**Fechar**para mudar o botão inicial do mouse da esquerda para a direita (tornando o mouse adequado para o uso na mão esquerda).
>
> Selecione **Applications** → **Accessories** → **Character Map** a partir da barra de menu principal, com o objetivo de inserir um caractere especial ao arquivo **gedit**. Em seguida, selecione **Search** → **Find…** a partir da barra do menu **Character Map**, digite o nome do caractere no campo **Search** e clique em **Next**. O caractere pesquisado aparecerá destacado no **Character Table**. Clique duas vezes no caractere destacado para posicioná-lo no campo **Text to copy** e clique no botão **Copy**. Retorne ao seu documento e selecione **Edit** → **Paste** a partir da barra do menu **gedit**.

O texto acima inclui nomes de aplicativos, nomes de menu e itens de todo o sistema, nomes de menu específicos do aplicativo, e botões e textos encontrados na Interface Gráfica GUI, todos apresentados em Negrito Proporcional (Proportional Bold) e todos diferenciados de acordo com o contexto.

***Itálico em Negrito de Espaço Único (Mono-spaced Bold Italic)*** ou ***Itálico em Negrito Proporcional (Proportional Bold Italic)***

Sendo o Negrito Espaço Único (Mono-spaced Bold) ou Negrito Proporcional (Proportional Bold), os itálicos extras indicam textos substituíveis ou variáveis. O Itálico denota o texto que você não inseriu literalmente ou textos exibidos que mudam dependendo das circunstâncias. Por exemplo:

> Para conectar-se à uma máquina remota usando o ssh, digite **ssh *nome do usuário*@*domain.name*** na janela de comandos. Por exemplo, considere que a máquina remota seja **example.com** e seu nome de usuário nesta máquina seja john, digite **ssh john@example.com**.
>
> O comando **mount -o remount *file-system*** remonta o sistema de arquivo nomeado. Por exemplo, para remontar o sistema de arquivo **/home**, o comando é **mount -o remount /home**.
>
> Para ver a versão de um pacote instalado, use o comando **rpm -q *package***. Ele retornará um resultado como este: ***package-version-release***.

Perceba as palavras em negrito e itálico acima - username, domain.name, file-system, package, version e release. Cada palavra é um espaço reservado, tanto para o texto que você insere quando emitindo um comando ou para textos exibidos pelo sistema.

Além de uso padrão para apresentar o título de um trabalho, os itálicos denotam a primeira vez que um termo novo e importante é usado. Por exemplo:

> O Publican é um sistema de publicação do *DocBook*.

## 1.2. Convenções de Pull-Quote

Resultado de terminal e listagem de código fonte são definidos visualmente com base no contexto.

O resultado enviado à um terminal é configurado em **Romano de Espaço Único (Mono-spaced Roman)** e apresentado assim:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

As listas de código fonte também são configuradas em **Romano de Espaço Único (Mono-spaced Roman)**, porém são apresentadas e realçadas como a seguir:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                    struct kvm_assigned_pci_dev *assigned_dev)
{
          int r = 0;
          struct kvm_assigned_dev_kernel *match;

          mutex_lock(&kvm->lock);

          match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                        assigned_dev->assigned_dev_id);
          if (!match) {
                    printk(KERN_INFO "%s: device hasn't been assigned before, "
                      "so cannot be deassigned\n", __func__);
                    r = -EINVAL;
                    goto out;
          }

          kvm_deassign_device(kvm, match);

          kvm_free_assigned_device(kvm, match);

out:
          mutex_unlock(&kvm->lock);
          return r;
}
```

## 1.3. Notas e Avisos

E por fim, usamos três estilos visuais para chamar a atenção para informações que possam passar despercebidas.

### Nota

Uma nota é uma dica ou símbolo, ou ainda uma opção alternativa para a tarefa em questão. Se você ignorar uma nota, provavelmente não resultará em más consequências, porém poderá deixar passar uma dica importante que tornará sua vida mais fácil.

### Importante

Caixas importantes detalham coisas que são geralmente fáceis de passarem despercebidas: mudanças de configuração que somente se aplicam à sessão atual, ou serviços que precisam ser reiniciados antes que uma atualização seja efetuada. Se você ignorar estas caixas importantes, não perderá dados, porém isto poderá causar irritação e frustração.

### Atenção

Um Aviso não deve ser ignorado. Se você ignorar avisos, muito provavelmente perderá dados.

## 2. Obtendo Ajuda e Fornecendo Comentários

### 2.1. Você precisa de ajuda?

Se encontrar dificuldades com o procedimento descrito nesta documentação, você pode encontrar ajuda visitando o Portal do Cliente da Red Hat em http://access.redhat.com. Você poderá realizar o seguinte através do portal do cliente:

- realize buscas ou navegue através da base de conhecimento dos artigos de suporte técnico sobre os produtos da Red Hat.
- insira o caso de suporte nos Serviços de Suporte Global da Red Hat (GSS).
- acesso a outra documentação da Red Hat.

A Red Hat acomoda um grande número de listas de correio eletrônico para discussão de software e tecnologias da Red Hat. Você pode encontrar uma lista contendo as listas públicas disponíveis em https://www.redhat.com/mailman/listinfo. Clique no nome de qualquer lista para subscrever àquela lista ou acessar o histórico das listas.

### 2.2. Nós precisamos do seu Comentário!

Se você encontrar erro ortográfico neste manual, ou se você tem uma sugestão para melhoramento deste manual, nós gostaríamos muito de ouvir sua opinião! Por favor submita um relatório em Bugzilla:http://bugzilla.redhat.com/ contra o produto **Red Hat Enterprise Linux 6.**.

Ao submeter um relatório de erro, certifique-se de mencionar a identificação do manual:*doc-Performance_Tuning_Guide*.

Se você tiver uma sugestão para o melhoramento deste documento, por favor tente ser o mais específico possível em sua descrição. Caso você tenha encontrado um erro, por favor inclua o número da seção e alguns detalhes a respeito do texto para a nossa melhor identificação.

# Capítulo 1. Visão Geral

O *Guia de Ajuste de Desempenho* é uma referência compreensiva sobre configuração e otimização do Red Hat Enterprise Linux. Embora este lançamento também contenha informações sobre as capacidades de desempenho do Red Hat Enterprise Linux 5, todas as instruções fornecidas aqui são específicas para o Red Hat Enterprise Linux 6.

Este livro é dividido em capítulos que discutem subsistemas específicos no Red Hat Enterprise Linux. O *Guia de Ajuste de Desempenho* foca em três temas principais por subsistema:

**Recursos**

Cada capítulo de subsistema descreve recursos de desempenho únicos (ou implementados de forma diferente) do Red Hat Enterprise Linux 6. Estes capítulos discutem as atualizações do Red Hat Enterprise Linux 6 que aprimoraram o desempenho de subsistemas específicos ao longo do Red Hat Enterprise Linux 5.

**Análise**

O livro também enumera os indicadores de desempenho para cada subsistema específico. Valores típicos para estes indicadores são descritos no contexto de serviços específicos, ajudando-o a entender seu significado na produção de sistemas do mundo real.

Além disso, o *Guia de Ajuste de Desempenho* também mostra formas diferentes de recuperar os dados de desempenho (ou seja, perfil) para um subsistema. Note que algumas das ferramentas de perfil demonstradas aqui são documentadas em outros locais em mais detalhes.

**Configuração**

Talvez a informação mais importante neste livro seja as instruções sobre como ajustar o desempenho de um subsistema específico no Red Hat Enterprise Linux 6. O *Guia de Ajuste de Desempenho* explica como ajustar de forma fina um subsistema do Red Hat Enterprise Linux 6 para serviços específicos.

Tenha em mente que ao ajustar um desemepnho de subsistema específico, você pode afetar o desempenho de outro, as vezes adverso. A configuração padrão do Red Hat Enterprise Linux 6 é adequada para a *maioria* dos seviços executando sob cargas *moderadas*.

Os procedimentos enumerados no *Guia de Ajuste de Desempenho* foram testados extensivamente pelos engenheiros da Red Hat, em laboratórios e em campo. No entanto, a Red Hat recomenda que você teste corretamente todas as configurações programadas em um ambiente de teste seguro antes de aplicá-lo para um ambiente de produção. Você também deve fazer backup de todos os dados e informações de configurações antes de ajustar seu sistema.

## 1.1. Público Alvo

Este livro é adequado para dois tipos de leitores:

**Analistas de Sistema/Negócios**

Este livro enumera e explica os recursos de desempenho do Red Hat Enterprise Linux 6 a um nível elevado, fornecendo informações suficientes sobre como os subsistemas para executar cargas de trabalho específicas (tanto por padrão e quando otimizado). O nível de detalhamento utilizado na descrição dos recursos de desempenho do Red Hat Enterprise Linux 6 ajuda

clientes em potencial e engenheiros de vendas a entenderem a adequação desta plataforma na prestação de serviços intensivos em recursos a um nível aceitável.

O *Guia de Ajuste de Desempenho* também fornece links para documentação mais detalhada em cada recurso onde seja possível. Neste nível de detalhes, os leitores podem entender estes recursos de desempenho suficientes para formar uma estratégia de alto nível ao implementar e otimizar o Red Hat Enterprise Linux 6. Isto permite que leitores desenvolvam *e* avaliem as propostas de infraestrutura.

Este nível de recurso focado de documentação é adequado para leitores com um alto nível de conhecimento do subsistema Linux e redes de nível corporativo.

**Administrador de Sistemas**

Os procedimentos enumerados neste livro são adequados aos administradores de sistema com o nível de qualificações RHCE [1] ( ou equivalente, ou seja, de 3 à 5 anos de experiência na implantação e gerenciamento do Linux). O *Guia de Ajuste de Desempenho* foca em prover o máximo de detalhes possível sobre os efeitos de cada configuração, ou seja, descrevendo qualquer compensação de desempenho que possa ocorrer.

A qualificação subjacente no ajuste de desempenho não depende de como analisar e ajustar um subsistema. Ao invés disso, um administrador de sistema adepto ao ajuste de desempenho, sabe como balancear e otimizar um sistema Red Hat Enterprise Linux 6 system *para propósitos específicos*. Isto *também* significa saber quais penalidades de ajuste e compensações são aceitáveis para implementar uma configuração criada para aumentar um desempenho de subsistema específico.

## 1.2. Escalabilidade Horizontal

Os esforços da Red Hat em aprimorar o desempenho do Red Hat Enterprise Linux 6 foca na *escalabilidade*. Os recursos de aumento de desempenho são avaliados primeiramente baseados em quanto afetam o desempenho de plataforma em áreas diferentes de um spectrum de carga de trabalho — ou seja, a partir de servidores de Web solitários para a mainframe do servidor.

Concentrar-se em escalabilidade permite que o Red Hat Enterprise Linux mantenha a sua versatilidade em diferentes tipos de cargas de trabalho e propósitos. Ao mesmo tempo, isso significa que a medida que sua empresa cresce e suas escalas de carga de trabalho aumentam,a re-configuração de seu ambiente de servidor é menos proibitiva (em termos de custo e mão de obra) e mais intuitiva.

A Red Hat faz melhorias no Red Hat Enterprise Linux, tanto para *escalabilidade horizontal* quanto para *escalabilidade vertical,* , no entanto, a escalabilidade horizontal é o caso de uso mais geralmente aplicável. A idéia por trásda escalabilidade horizontal é a utilização de vários *computadores padrão* para distribuir cargas de trabalho pesadas, a fim de melhorar o desempenho e a confiabilidade.

Em um farm de servidor típico, esses computadores padrão vêm na forma de servidores montados em rack 1U e servidores blade. Cada computador padrão pode ser do tamanho de um sistema de dois soquetes simples, embora algumas farms de servidor utilizam sistemas grandes, com mais soquetes. Algumas redes de nível empresarial misturam sistemas grandes e pequenos, em tais casos, os grandes sistemas são servidores de alto desempenho (por exemplo, servidores de banco de dados) e os pequenos são servidores de aplicação dedicados (por exemplo, servidores de web ou e-mail).

Este tipo de escalabilidade simplifica o crescimento de sua infra-estrutura de TI: um negócio de tamanho médio com uma carga adequada pode só precisar de dois servidores de caixa de pizza para

atender todas as suas necessidades. À medida que a empresa contrata mais pessoas, expande suas operações, aumenta seu volume de vendas e assim por diante, as suas necessidades de TI aumentam em volume e complexidade. Escalabilidade horizontal permite que a TI simplesmente implante máquinas adicionais com (principalmente) configurações idênticas, como os seus antecessores.

Para resumir, a escalabilidade horizontal adiciona uma camada de abstração que simplifica a administração de hardware do sistema. Ao desenvolver a plataforma Red Hat Enterprise Linux para escalar horizontalmente, o aumento da capacidade e o desempenho dos serviços de TI podem ser tão simples como a adição de novas máquinas, facilmente configurados.

### 1.2.1. Computação Paralela

Os usuários se beneficiam de escalabilidade horizontal do Red Hat Enterprise Linux e não apenas porque ele simplifica a administração de hardware do sistema, mas também porque a escalabilidade horizontal é uma filosofia de desenvolvimento adequado, dadas as tendências atuais no avanço hardware.

Considere o seguinte: as aplicações corporativas mais complexas têm milhares de tarefas que devem ser executadas simultaneamente, com diferentes métodos de coordenação entre as tarefas. Enquanto primeiros computadores tinham um processador single-core para conciliar todas essas tarefas, praticamente todos os processadores disponíveis hoje têm múltiplos núcleos. Efetivamente, os computadores modernos colocam múltiplos núcleos em um único soquete, fazendo até mesmo desktops soquetes individuais ou laptops de sistemas multi-processador.

Desde 2010, o padrão Intel e AMD estão disponíveis com 2-16 núcleos. Esses processadores são prevalentes em caixa de pizza ou servidores blade, que agora podem conter até 40 núcleos. Estes sistemas de baixo custo e de alto desempenho trazem capacidades do sistema de grande porte e características para o mainstream.

Para conseguir o melhor desempenho e a utilização de um sistema, cada núcleo tem de ser mantido ocupado. Isso significa que 32 tarefas distintas devem estar em execução para tirar proveito de um servidor blade de 32-núcleos. Se um chassis lâmina contém dez dessas lâminas 32 núcleos, toda a configuração pode processar um mínimo de 320 tarefas simultaneamente. Se essas tarefas fazem parte de um único trabalho, elas devem ser coordenados.

Red Hat Enterprise Linux foi desenvolvido para se adaptar bem às tendências de desenvolvimento de hardware e garantir que as empresas podem se beneficiar totalmente deles. Seção 1.3, “Sistemas Distribuídos” explora as tecnologias que permitem escalabilidade horizontal do Red Hat Enterprise Linux em maior detalhes.

## 1.3. Sistemas Distribuídos

Para realizar plenamente a escalabilidade horizontal, a Red Hat Enterprise Linux usa muitos componentes da *computação distribuída* . As tecnologias que compõem a computação distribuída são divididas em três camadas:

**Comunicação**

Escalabilidade horizontal requer muitas tarefas a serem executadas simultaneamente (em paralelo). Como tal, estas tarefas devem ter *comunicação de interprocesso* para coordenar seu trabalho. Além disso, uma plataforma com escalabilidade horizontal deve ser capaz de compartilhar tarefas entre vários sistemas.

**Armazenamento**

Armazenamento via discos locais não é suficiente para enfrentar as exigências de

escalabilidade horizontal. Será necessária alguma forma de armazenagem compartilhada ou distribuída, uma com uma camada de abstração que permite que a capacidade de um único volume de armazenamento cresça de forma integrada com a adição de um novo hardware de armazenamento.

**Gerenciamento**

O dever mais importante na computação distribuída é a camada de *gestão*. Esta camada de gerenciamento coordena todos os componentes de software e hardware, gestão eficiente de comunicação, armazenamento e uso de recursos compartilhados.

As seguintes seções descrevem as tecnologias dentro de cada camada em mais detalhes.

### 1.3.1. Comunicação

A camada de comunicação assegura o transporte de dados, e é composto de duas partes:

- Hardware
- Software

A forma mais simples (e mais rápida) de sistemas múltiplos se comunicarem é através de *memória compartilhada*. Isto possibilita o uso de leitura de memória familiar/operações de gravação; memória compartilhada possui uma largura de banda alta, baixa latência e baixa sobrecarga de leitura de memória comum/operações de gravação.

#### Ethernet

A forma mais comum de comunicação entre computadores é sob Ethernet. Hoje, *Gigabit Ethernet* (GbE) é fornecido por padrão nos sistemas, e a maioria dos servidores incluem 2-4 portas Gigabit Ethernet. GbE fornece boa largura de banda e latência. Esta é a base da maioria dos sistemas distribuídos em uso nos dias de hoje. Mesmo quando os sistemas incluem hardware de rede mais rápida, ainda é comum o uso de GbE para uma interface de gerenciamento dedicado.

#### 10GbE

*Ten Gigabit Ethernet* (10GbE) está crescendo rapidamente na aceitação para servidores de alto nível e até mesmo servidores de nível médio. 10GbE fornece dez vezes a largura de banda de GbE. Uma de suas principais vantagens é com processadores multi-core modernos, onde se restabelece o equilíbrio entre comunicação e computação. Você pode comparar um sistema de núcleo único com GbE com um sistema de oito núcleos usando 10GbE. Usado desta forma, o 10GBE é especialmente valioso para manter o desempenho geral do sistema e evitar afunilamento de comunicação.

Infelizmente, o 10GBE é caro. Enquanto o custo de placas de rede 10GbE desceu, o preço da interconexão (especialmente de fibra ótica) permanece elevado, e interruptores de rede 10GbE são extremamente caros. Podemos esperar que esses preços caiam ao longo do tempo, mas 10GbE hoje é mais fortemente utilizado em suporte principal de sala de servidores e aplicativos de desempenho crítico.

#### Infiniband

Infiniband oferece um desempenho ainda mais alto do que 10GbE.Além das conexões de rede TCP/IP e UDP utilizadas com o Ethernet, o Infiniband também suporta comunicação de memória compartilhada. Isto permite que o Infiniband funcione entre sistemas via *acesso de memória direto remoto* (RDMA).

O uso de RDMA permite que o Infiniband mova dados diretamente entre os sistemas sem a sobrecarga

de conexões TCP / IP ou socket. Por sua vez, reduz a latência, o que é fundamental para algumas aplicações.

Infiniband é mais comumente usado em aplicativos *Computação de Alto Desempenho Técnico* (HPTC), que requerem elevada largura de banda, baixa latência e baixo custo operacional. Muitas aplicações de supercomputação beneficiam-se com isso, a ponto de que a melhor maneira de melhorar o desempenho é através do investimento em Infiniband ao invés de processadores mais rápidos ou mais de memória.

#### RoCCE

*RDMA over Ethernet* (RoCCE) implementa as comunicações em estilo do Infinibands (including RDMA) sob uma infraestrutura 10GbE. Considerando as melhorias de custo associada ao volume crescente dos produtos 10GbE, é razoável esperar uso mais amplo do RDMA e RoCCE em uma classe ampla em uma classe de sistemas e aplicativos.

Cada um destes métodos de comunicação é totalmente suportado pelo Red Hat for use with Red Hat Enterprise Linux 6.

### 1.3.2. Armazenamento

Um ambiente que usa computação distribuída usa várias instâncias de armazenamento compartilhado. Isto pode significar uma entre duas coisas:

- Dados de armazenamento de sistemas múltiplos em um único local
- uma unidade de armazenamento (ex.: um volume) composto por equipamentos de armazenamento múltiplos.

O exemplo mais conhecido de armazenamento é o disco rígido local montado em um sistema. Isto é apropriado para operações de TI, onde todos os aplicativos são hospedados em um hospedeiro, ou até mesmo um pequeno número de hospedeiros. No entanto, como as escalas de infra-estrutura para dezenas ou mesmo centenas de sistemas, gestão de tantos discos de armazenamento locais torna-se difícil e complicado.

Armazenamento distribuído adiciona uma camada para facilitar e automatizar a administração de hardware de armazenamento, como as escalas de negócios. Ter vários sistemas que compartilham diversos casos de armazenamento, reduz o número de dispositivos que o administrador precisa gerenciar.

Consolidar as capacidades de armazenamento de vários dispositivos de armazenamento em um volume ajuda os usuários e administradores. Este tipo de armazenamento distribuído fornece uma camada de abstração para pools de armazenamento: os usuários vêem uma única unidade de armazenamento, o que um administrador pode facilmente fazer crescer, adicionando mais hardware. Algumas tecnologias que permitem armazenamento distribuído também oferecem benefícios adicionais, tais como failover e vários caminhos.

#### NFS

O *Network File System* (NFS) permite que múltiplos servidores ou usuários montem e utilizem a mesma instância de armazenamento remoto via TCP ou UDP. O NFS é geralmente utilizado para manter dados compartilhados por diversos aplicativos. É também conveniente para armazenamento em massa com uma grande quantidade de dados.

#### SAN

*Rede de Área de Armazenamento* (SANs) usa o Canal de Fibra ou o protocolo ou iSCSI para fornecer acesso remoto para armazenamento. A infraestrutura do Canal de Fibra, (tal como o adaptadores do

bus do host Canal de Fibra, interruptores e matrizes de armazenamento) combina com o alto desempenho, largura de banda alta e enorme armazenamento. O SANs separa o armazenamento do processamento, provando flexibilidade considerável na criação do sistema.

As outras vantagens principais do SANs é que eles fornecem um ambiente de gerenciamento para realizar tarefas grandes administrativas do hardware de armazenamento . Estas tarefas incluem:

- Controlando acesso ao armazenamento
- Gerenciamento de grande quantidade de dados
- Sistemas de Provisionamento
- Fazendo um backup e replicando dados
- Tirando snapshots
- Suportando failover de sistema
- Garantindo a integridade de dados
- Migrando dados

#### GFS2

O sistema de arquivo da Red Hat *Global File System 2* (GFS2) fornece vários recursos especializados. A função básica do GFS2 é fornecer um único sistema de arquivos, incluindo o concorrente de leitura / gravação de acesso, compartilhados entre vários membros de um cluster. Isto significa que cada membro do cluster vê exatamente os mesmos dados "no disco" no sistema de arquivos GFS2.

O GFS2 permite que todos os sistemas possuam acesso simultâneo ao "disco". Para manter a integridade de dados, o GFS2 utiliza um *Gerenciador de Bloqueio Distribuído* (DLM), o qual só permite que um sistema grave em um local específico por vez.

GFS2 se adequa bem especialmente aos aplicativos failover que requerem uma alta disponibilidade de armazenamento.

Para mais informações sobre o GFS2, consulte o *Global File System 2*. Para mais informações futuras sobre o armazenamento em geral, consulte o *Guia de Administração do Armazenamento*. Ambos estão disponíveis em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

### 1.3.3. Redes Convergidas

A Comunicação sob a rede é geralmente feita através da Ethernet, com o tráfego de armazenamento utilizando um ambiente de Canal de Fibra SAN. É comum ter uma rede dedicada ou link em série para o gerenciamento de sistemas e talvez até *heartbeat* [2]. Como resultado, um servidor único está geralmente em diversas redes.

Fornecer diversas conexões em cada servidor é caro, pesado e complexo de se gerenciar. Isto criou a necessidade de uma forma de consolidar todas as conexões em uma. O *Fibre Channel over Ethernet* (FCoE) e e *Internet SCSI* (iSCSI) se referem à esta necessidade.

#### FCoE

Com o FCoE, comandos de canal de fibra padrão e pacotes de dados são transportados através de uma infra-estrutura física de 10GbE através de uma única *placa de rede convergente* (CNA). Tráfego de ethernet padrão TCP/IP e operações de armazenamento de canal de fibra podem ser transportados através do mesmo link. FCoE usa um cartão de interface de rede física (e um único cabo) pararedes lógicas múltiplas/conexões de armazenamento.

FCoE oferece as seguintes vantagens:

**Número reduzido de conexões**

FCoE reduz o número de conexões de rede a um servidor pela metade. Você ainda pode optar por fazer várias conexões para o desempenho ou a disponibilidade, no entanto, uma única conexão fornece armazenamento e conectividade de rede. Isso é especialmente útil para os servidores de caixa de pizza e servidores blade, já que ambos têm um espaço muito limitado para os componentes.

**Custo mais baixo**

Reduzir número de conexões, imediatamente, significa número reduzido de cabos, interruptores e outros equipamentos de rede. A história da Ethernet também apresenta grandes economias de escala, o custo das redes cai drasticamente quando o número de dispositivos no mercado vai de milhões a bilhões, como foi visto na queda do preço de 100Mb Ethernet e dispositivos Gigabit Ethernet.

Da mesma forma, o 10GbE também se torna mais barato a medida que os negócios se adaptam ao seu uso. Como o hardware CNA, ele é integrado em um único chip, o uso espalhado também aumentará seu volume no mercado, o qual resultará em baixa de preço de forma significativa ao longo do tempo.

### iSCSI

*Internet SCSI* (iSCSI) é outro tipo de protocolo de rede convergida; é uma alternativa para o FCoE. Como o canal de fibra, o canal iSCSI fornece armazenamento de nível de bloco sob a rede. No entanto, o iSCSI não fornece um ambiente de gerenciamento completo. A principal vantagem do iSCSI sob o FCoE é que o iSCSI fornece muito da capacidade e flexibilidade do canal de fibra, mas com um custo menor.

---

[1] Engenheiro certificado da Red Hat. Para mais informações, consulte o http://www.redhat.com/training/certifications/rhce/.

[2] *Heartbeat* é a troca de mensagens entre sistemas para garantir que cada um ainda está funcionando. Se um sistema "perde o heartbeat" entende-se que ele falhou e está fechado, com outro sistema tomando a liderança.

# Capítulo 2. Recursos de Desempenho do Red Hat Enterprise Linux 6

## 2.1. Suporte de 64-bit

Red Hat Enterprise Linux 6 suporta processadores de 64 bits; estes processadores podem utilizar teoreticamente até 16 *exabytes* de memória. Como a Disponiblidade Geral (GA - General Availability), o Red Hat Enterprise Linux 6 é testado e certificado para suportar até 8TB de memória física.

O tamanho da memória suportada pelo Red Hat Enterprise Linux 6 deve crescer durante diversas atualizações menores, como a Red Hat continua a apresentar e aprimorar mais recursos que habilitam o uso de blocos de memória grandes. Exemplos de tais melhorias (desde o Red Hat Enterprise Linux 6 GA) são:

- Huge pages e transparent huge pages
- Melhorias de Acesso de Memória Não Uniforme

Estas melhorias são descritas em mais detalhes nas seções que se seguem.

### Huge pages e transparent huge pages

A implementação do *huge pages* no Red Hat Enterprise Linux 6 permite que o sistema gerencie o uso de memória de forma eficiente em diferentes cargas de trabalho de memória. Huge pages usa de forma dinâmica as páginas de 2 MB comparado ao tamanho de página padrão de 4 KB, permitindo aplicativos escalar bem a partir do processamento de gigabytes e até terabytes de memória.

As páginas enormes (huge pages) são difíceis de criar, gerenciar e utilizar manualmente. Para resolver este problema o Red Hat Enterprise 6 também apresenta o uso do *transparent huge pages* (THP). O THP gerencia automaticamente muitas das complexidades envolvidas no uso das páginas enormes.

Para mais informações sobre huge pages e THP, consulte o Seção 5.2, “Huge Pages e Transparent Huge Pages”.

### Melhorias do NUMA

Muitos dos novos sistemas suportam agora *Non-Uniform Memory Access* (NUMA). NUMA simplifica o desenho e criação de hardware para sistemas de grande porte, no entanto, ele também adiciona uma camada de complexidade para o desenvolvimento de aplicativos. Por exemplo, NUMA implementa a memória local e remota, onde a memória remota pode levar muito mais tempo para acessar a memória local. Este recurso (entre outros) possuem muitas implicações de desempenho que os sistemas operacionais de impacto, aplicativos e configurações do sistema devem ser implantados.

Red Hat Enterprise Linux 6 é melhor otimizado para uso do NUMA, graças a vários recursos adicionais que ajudam a gerenciar usuários e aplicações em sistemas NUMA. Esses recursos incluem a afinidade da CPU, fixação de CPU (cpusets), numactl e grupos de controle, que permitem que um processo (afinidade) ou aplicação (fixação) para "conectar" a uma CPU específica ou conjunto de processadores.

Para mais informações sobre o suporte do NUMA no Red Hat Enterprise Linux 6, consulte o Seção 4.1.1, “CPU e a Topologia NUMA ”.

## 2.2. Ticket Spinlocks

Uma parte chave de qualquer design de sistema é assegurar que um processo não altere a memória utilizada pelo outro processo. As mudanças de dados que não podem ser controladas em memória

podem resultar em danos de dados e travamento de sistema. Para evitar isto, o sistema operacional permite que um processo trave uma parte da memória, realize a operação e destrave ou "libere" a memória.

Uma implementação comum de trava de memória é através do *spin locks*, que permite que um processo continue verificando para ver se uma trava está disponível e pegá-la assim que estiver. Caso exista processos múltiplos competindo por uma mesma trava, a primeira requisição após ter sido liberada ganha. Quando todos os processos possuirem o mesmo acesso à memória, a situação será "justa" e funcionará bem.

Infelizmente, em um sistema NUMA, nem todos os processos têm igualdade de acesso aos bloqueios. Processos no mesmo nó NUMA como o bloqueio, possuem uma vantagem injusta na obtenção do bloqueio. Processos sobre nós NUMA remotos experimentam a falta de bloqueio e desempenho degradado.

Para solucionar este problema, o Red Hat Enterprise Linux implementou o *ticket spinlocks*. Este recurso adiciona um mecanismo de fila de reserva à trava, permitindo que *todos* os processos obtenham uma trava na ordem com a qual requisitarem -na. Isto elimina os problemas de tempo e vantagens injustas nas requisições de bloqueio.

Embora o ticket spinlock possua mais cabeçalho do que um spinlock comum, ele escala melhor e provê melhor desempenho em sistemas NUMA.

## 2.3. Estrutura da Lista Dinâmica

O sistema operacional requer um conjunto de informações sobre cada processador do sistema. No Red Hat Enterprise Linux 5, este conjunto de informações foi atribuído a uma matriz de tamanho fixo na memória. Informações sobre cada processador individual foi obtido por indexação nessa matriz. Este método é rápido, fácil e simples para sistemas que continham relativamente poucos processadores.

No entanto, como o número de processadores de um sistema cresce, este método produz uma sobrecarga significativa. Como a matriz de tamanho fixo na memória é um recurso único, compartilhado, pode se afunilar à medida que mais processadores tentam acessá-lo ao mesmo tempo.

Para resolver isso, o Red Hat Enterprise Linux 6 utiliza uma *estrutura de lista dinâmica* para obter informações do processador. Isso permite que a matriz utilizada para informações sobre o processador seja alocada dinamicamente: se há apenas oito processadores no sistema, então apenas oito entradas são criadas na lista. Se existirem processadores 2048, então 2048 são criadas entradas bem.

A estrutura de lista dinâmica permite bloqueio mais refinado. Por exemplo, se a informação precisa ser atualizado ao mesmo tempo para os processadores de 6, 72, 183, 657, 931 e 1546, isso pode ser feito com maior paralelismo. Situações como esta, obviamente, ocorrem com muito mais freqüência em sistemas de grande porte e de alto desempenho do que em sistemas pequenos.

## 2.4. Tickless Kernel

Em versões anteriores do Red Hat Enterprise Linux, o kernel utilizava um mecanismo baseado em tempo, que produzia continuamente uma interrupção de sistema. Durante cada interrupção, o sistema *polled*; ou seja, ele verificada para ver se havia algum trabalho a ser realizado.

Dependendo da configuração, esta interrupção de sistema ou *timer tick* ocorria diversas vezes por segundo. Isto acontecia a cada segundo, sem considerar a carga de trabalho do sistema. Em um sistema carregado de forma leve, isto impacta o *consumo de energia*, prevenindo o processador de utilizar os estados de suspensão de forma efetiva. O sistema utiliza o mínimo de energia quando está

no estado de suspensão.

A forma mais efetiva de um sistema operar é funcionar o mais rápido possível, vá para o estado de suspensão mais profundo possível, e deixe-o suspenso durante o máximo de tempo possível. Para implementar isto, o Red Hat Enterprise Linux 6 utiliza um *tickless kernel*. Com isto, o relógio de interrupção foi removido do loop ocioso, transformando o Red Hat Enterprise Linux 6 em um ambiente liderado completamente pela interrupção.

O kernel tickless permite que o sistema vá para o estado de supensão profunda durante os horários em que se encontra em estado ocioso, e responde prontamente quando há algum trabalho a ser realizado.

Para maiores informações consulte o *Guia de Gerenciamento de Energia*, available from http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

## 2.5. Grupos de Controle

Red Hat Enterprise Linux oferece muitas opções úteis para ajuste do desempenho. Sistemas de grande porte, que escalam em centenas de processadores, podem ser ajustados para fornecer excelente desempenho. Mas ajuste desses sistemas requer considerável experiência e uma carga de trabalho bem definido. Quando grandes sistemas eram caros e poucos em número, era aceitável dar-lhes um tratamento especial. Agora que esses sistemas são mainstream, são necessários instrumentos mais eficazes.

Para complicar ainda mais as coisas, os sistemas mais poderosos estão sendo usados agora para a consolidação de serviço. As cargas de trabalho, que eram executadas em 4-8 servidores mais antigos são agora colocadas num único servidor. E, como discutido anteriormente Seção 1.2.1, “Computação Paralela”, muitos sistemas de médio alcance nos dias de hoje contêm mais núcleos do que máquinas antigas de alto desempenho.

Muitos aplicativos modernos são projetados para processamento paralelo, usando vários segmentos ou processos para melhorar o desempenho. No entanto, alguns aplicativos podem fazer uso efetivo de mais de oito threads. Por isso, várias aplicações normalmente precisam ser instaladas em um sistema de 32 CPU para maximizar a capacidade.

Considere a situação: pequenos sistemas convencionais de baixo custo já estão em paridade com o desempenho das máquinas antigas caras e de alto desempenho. Máquinas de alto desempenho mais baratas ofereceram aos arquitetos de sistemas a capacidade de consolidar mais serviços em menos máquinas.

No entanto, alguns recursos (tal como E/S e comunicação de rede) são compartilhadas e não crescem tão rapidamente quanto a conta de CPU. Como tal, um sistema que acomoda aplicativos múltiplos podem experienciar desempenho geral degradado quando um aplicativo monopoliza demais um único recurso.

Para resolver isso, o Red Hat Enterprise Linux 6 suporta agora os *grupos de controle* (cgroups). Os Cgroups permitem que os administradores aloquem recursos para tarefas específicas conforme necessário. Isto significa, por exemplo, ser capaz de alocar 80% dos quatro CPUs, 60GB de memória, e 40% de E/S de disco para um aplicativo de banco de dados. O aplicativo da Web em execução no mesmo sistema pode ter recebido duas CPUs, 2GB de memória, e 50% da largura de banda de rede disponível.

Como resultado, o banco de dados e os aplictivos da Web fornecem bom desempenho, pois o sistema evita que ambos consumam recursos de sistema excessivamente. Além disso, muitos aspectos do cgroups são *auto-ajustáveis*,permitindo que o sistema responda de forma adequada à mudanças na

carga de trabalho.

Um cgroup possui dois componentes principais:

- Uma lista de tarefas atribuídas ao cgroup
- Recursos alocados a estas tarefas

As tarefas atribuídas ao cgroup executam *dentro* do cgroup. Qualquer tarefa filho que eles gerem também executam dentro do cgroup. Isto permite que um administrador gerencie um aplicativo todo como uma unidade única. Um administrador também pode configurar alocações para os seguintes recursos:

- CPUsets
- Memória
- E/S
- Rede (largura de banda)

Dentro das CPUsets, os cgroups permitem que administradores configurem o número de CPUs, afinidades para CPUs específicas ou nós [3], e a quantia de tempo da CPU utilizada por um conjunto de tarefas. O uso dos cgroups para configurar o CPUsets é crucial para garantir um desempenho bom geral, prevenindo um aplicativo de consumir recursos de forma excessiva no custo de outras tarefas enquanto garante simultaneamente que o aplicativo não falte durante o tempo de CPU.

A largura de banda de E/S e largura de banda de rede são gerenciadas por outros controladores de recursos. Novamente, os controladores de recursos permitem que vocÊ determine quanto de largura de banda as tarefas no cgroup poderão consumir, e garantir que as tarefas no cgroup não consumam recursos excessivamente nem fique sem recursos.

Os Cgroups permitem que os administradores definam e aloquem, em um alto nível, os recursos de sistemas que diversos aplicativos precisam (e continuarão) a consumir. O sistema então gerencia e balnceia automaticamente os diversos aplicativos, entregando um bom desempenho já esperado e otimizando o desempenho do sistema geral.

Para mais informações sobre como utilizar os grupos de controle, consulte o *Guia de Gerenciamento de Recursos*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

## 2.6. Melhorias de Armazenamento de Sistema de Arquivo

O Red Hat Enterprise Linux 6 também apresenta diversas melhorias no gerenciamento do armazenamento e sistema de arquivo. Dois avanços entre os mais notáveis nesta versão são o ext4 e suporte XFS. Para mais cobertura compreensiva das melhorias de desempenho relacionadas ao armazenamento e sistemas de arquivo, consulte o Capítulo 7, *Sistemas de Arquivos*.

### Ext4

Ext4 é o sistema de arquivos padrão para o Red Hat Enterprise Linux 6. É a quarta versão da geração da família de sistema de arquivos EXT, apoiando um tamanho máximo teórico do sistema de arquivos de 1 exabyte, e tamanho máximo de arquivo único de 16TB. Red Hat Enterprise Linux 6 suporta um tamanho máximo de arquivo do sistema de 16TB, e um único tamanho máximo de 16TB. Além de uma capacidade de armazenamento muito maior, ext4 também inclui várias novas funcionalidades, tais como:

- Metadados baseados em Extensão
- Alocação atrasada
- Diário do check-summing

Para mais informações sobre o sistema de arquivo ext4, consulte o Seção 7.3.1, “Sistema de Arquivo Ext4”.

### XFS

XFS é um sistema de arquivo de 64 bits robusto e maduro que suporta arquivos grandes e sistemas de arquivos em um único host. Este sistema de arquivos foi originalmente desenvolvido pela SGI, e tem uma histórico longo de execução em servidores extremamente grandes e diretrizes de armazenamento. As características XFS incluem:

- Alocação atrasada
- Inodes alocados de forma dinâmica
- Índice B-tree para escalabilidade de gerenciamento de espaço livre.
- Defragmentação online e crescente sistema de arquivo
- Algorítimos de leitura avançada de metadados sofisticados

Embora o XFS escale para exabytes, o tamanho máximo de sistema de arquivo XFS suportado pela Red Hat é de 100TB. Para mais informações sobre o XFS, consulte o Seção 7.3.2, “O Sistema de Arquivo XFS”.

### Drives de Inicialização Grandes

BIOS tradicional suporta um tamanho máximo do disco de 2.2TB. Os sistemas Red Hat Enterprise Linux 6 que utilizam o BIOS podem suportar discos maiores que 2.2TB usando uma nova estrutura de disco chamado *Partition Table global*(GPT). GPT só pode ser utilizado para discos de dados, ele não pode ser utilizada em unidades de inicialização com o BIOS e, portanto, os discos de inicialização podem ser de tamanho máximo de 2.2TB. A BIOS foi criado originalmente para o IBM PC, enquanto BIOS evoluiu consideravelmente para se adaptar ao hardware moderno, o *Unified Extensible Firmware Interface* (UEFI) foi projetado para suportar hardwares novos e emergentes.

O Red Hat Enterprise Linux 6 suporta o UEFI, que pode ser utilizado para substituir o BIOS (ainda suportado). Os sistemas que contém o UEFI e executam o Red Hat Enterprise Linux 6 permitem o uso do GPT e 2.2TB (e maiores) partições para ambas partições de inicialização e partição de dados.

**Importante - UEFI para sistemas de 32 bits x86**

Red Hat Enterprise Linux 6 não suporta o UEFI para sistemas 32-bit x86.

**Importante - UEFI para AMD64 e Intel 64**

Note que as configurações de inicialização do UEFI e BIOS diferem de maneira significante uma da outra. Portanto, o sistema instalado deve inicializar usando o mesmo firmware que era usado durante a instalação. Você não pode instalar o sistema operacional em um sistema que usa o BIOS e depois inicializar esta instalação em um sistema que usa o UEFI.
O Red Hat Enterprise Linux 6 suporta a versão 2.2 das especificações do UEFI. O Hardware que suporta a versão 2.3 das especificações do UEFI ou mais recentes, devem inicializar e operar com o Red Hat Enterprise Linux 6, mas a funcionalidade adicional definida por estas especificações mais recentes não estarão disponíveis. As especificações do UEFI estão disponíveis em http://www.uefi.org/specs/agreement/.

[3] Um nó é geralmente definido como um conjunto de CPUs ou núcleos dentro de um soquete.

# Capítulo 3. Monitorando e Analisando Desempenho de Sistema

Este capítulo introduz brevemente ferramentas que podem ser utilizadas para monitorar e analisar desempenho de sistema e aplicativo, e aponta situações nas quais as ferramentas podem ser úteis. Os dados coletados por cada ferramenta podem revelar funilamentos ou outros problemas de sistema que contribuem para um desempenho menos adequado.

## 3.1. O Sistema de Arquivos Proc

O **proc** "sistema de arquivos" é um diretório que contém uma hierarquia de arquivos que representam o estado atual do kernel do Linux. Ele permite que aplicativos e usuários acessem a visualização do kernel do sistema.

O **proc** diretório também contém informações sobre o hardware do sistema, e todos os processos em execução. A maioria destes arquivos são somente leitura, mas alguns arquivos (principalmente aqueles em **/proc/sys**) podem ser manipulados por usuários e aplicações para comunicar as alterações de configuração com o kernel.

Para maiores informações sobre como visualizar e editar arquivos no diretório **proc**, consulte o *Deployment Guide*, disponível a partir do http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

## 3.2. GNOME e Monitores de Sistema KDE

Os ambientes de desktop do GNOME e KDE possuem ferramentas de gráfico para assistí-lo no monitoramento e modificação do comportamento de seu sistema.

### Monitor de Sistema GNOME

O **GNOME System Monitor** exibe informações de sistema básicas e permite que você monitore processos de sistema e uso de sistema de arquivo ou recurso. Abra-o com o comando **gnome-system-monitor** no **Terminal**, ou clique em menu de **Applications** e selecione **System Tools** > **System Monitor**.

**Monitor de Sistema GNOME** possui quatro abas:

**Sistema**
: Exibe informações básicas sobre o hardware e software do computador

**Processos**
: Mostra processos ativos, e as relações entre os processos, bem como informações detalhadas sobre cada processo. Ele também permite que você filtre os processos apresentados, e executar determinadas ações nesses processos (iniciar, parar, matar, alterar a prioridade, etc.)

**Recursos**
: Exibe o uso de tempo de CPU atual, uso de memória e de espaço swap e uso de rede.

**Sistema de Arquivo**
: Lista todos os sistemas de arquivo montados junto com informações básicas sobre cada um, tal como tipo de sistema de arquivo, ponto de montagem e uso de memória.

Para mais informações sobre o **GNOME System Monitor**,consulte o menu **Help** no aplicativo ou no *Deployment Guide*, disponível a partir de http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

### Guarda de Sistema KDE

O **KDE System Guard** permite que você monitore a carga de sistema atual e processos que estejam em execução. Ele também deixa-o realizar ações e processos. Abra-o com o comando **ksysguard** no **Terminal**,ou clique em **Kickoff Application Launcher** and select **Applications** > **System** > **System Monitor**.

Existem duas abas em **KDE System Guard**:

**Tabela de Processo**
: Exibe uma lista de todos os processos em execução, em ordem alfabética por padrão. Você também pode classificar os processos por uma série de outras propriedades, incluindo o uso total da CPU, uso da memória física ou compartilhado, proprietário e prioridade. Você também pode filtrar os resultados visíveis, procurar processos específicos, ou realizar determinadas ações em um processo.

**Carga de Sistema**
: Exibe gráficos históricos de uso de CPU, memória e uso do espaço de troca e uso da rede. Passe o mouse sobre os gráficos para uma análise detalhada e as chaves de gráficos.

Para mais informações sobre o **KDE System Guard**, consulte o menu **Help** no aplicativo.

## 3.3. Ferramentas de Monitoramento de Linha de Comando Embutida

Além de ferramentas de monitoramento de gráficos, Red Hat Enterprise Linux oferece diversas ferramentas que podem ser usados para monitorar um sistema a partir da linha de comando. A vantagem destes instrumentos é que eles podem ser utilizadas fora do nível de execução 5. Esta seção descreve brevemente cada ferramenta, e sugere os fins para cada ferramenta.

### `top`

A ferramenta **top** fornece uma visão dinâmica em tempo real dos processos em um sistema em execução. Ela pode exibir uma variedade de informações, incluindo um resumo do sistema e as tarefas sendo gerenciadas pelo kernel do Linux. Ela também tem uma capacidade limitada para manipular processos. Tanto a sua operação quanto as informações que ela exibe são altamente configuráveis e quaisquer detalhes de configuração podem ser feitos para persistir nas reiinicializações.

Por padrão, os processos demonstrados são ordenados pela porcentagem do uso da CPU, fornecendo uma visão fácil dos processos que estão consumindo mais recursos.

Para mais detalhes sobre como utilizar o **top**, consulte esta página man: **`man top`**.

### `ps`

A ferramenta **ps** tira um snapshot de um grupo seleto de processos ativos. Por padrão, este grupo é

limitado a processos de usuários atuais e associados com o mesmo terminal.

Ele pode fornecer mais informações detalhadas sobre os processos além do **top**, mas não dinâmico.

Para mais informações detalhadas sobre como utilizar o **ps**, consulte a página man: **man ps**.

### vmstat

Os resultados do **vmstat** (Virtual Memory Statistics) reporta instantaneamente sobre os processos do seu sistema, memória e paginação, E/S de bloco e atividade de CPU.

No entanto não é dinâmico como **top**, você pode especificar um intervalo de amostra, o que o deixa observar atividade de sistema em tempo quase real.

Para mais informações detalhadas sobre como utilizar o **vmstat**, consulte a página man: **man vmstat**.

### sar

O **sar** (System Activity Reporter) coleta e reporta informações sobre as atividades de sistema de hoje. O resultado padrão cobre o uso da CPU de hoje em dez minutos de intervalo a partir do início do dia:

```
12:00:01 AM     CPU     %user     %nice    %system    %iowait    %steal     %idle
12:10:01 AM     all      0.10      0.00       0.15       2.96      0.00     96.79
12:20:01 AM     all      0.09      0.00       0.13       3.16      0.00     96.61
12:30:01 AM     all      0.09      0.00       0.14       2.11      0.00     97.66
...
```

Esta ferramenta é uma alternativa útil para tentar criar formulários periódicos em atividades de sistema através do **top** ou ferramentas semelhantes.

Para mais informações detalhadas sobre como utilizar o **sar**, consulte a página man: **man sar**.

## 3.4. Tuned e ktune

**Tuned** é um daemon que monitora e coleta de dados no uso de diversos componentes de sistemas, e utiliza esta informação para ajustar a configuração do sistema de forma dinâmica. Pode reagir a mudanças no uso da CPU e rede e ajustar configurações para aprimorar desempenho em dispositivos ativos ou reduzir o consumo de energia em dispositivos inativos.

Os parceiros acompanhantes **ktune** com a ferramenta **tuned-adm** para fornecer diversos perfis de ajuste que sejam pré-configurados para aprimorar o desempenho e reduzir o consumo de energia em diversos casos de uso específicos. Editar estes perfis ou criar novos perfis para criar soluções de desempenho feitas especialmente para seu ambiente.

Os perfis fornecidos como parte do **tuned-adm** incluem:

***default***

O perfil de economia de energia padrão. Este é o perfil de economia de energia mais básico. Ele permite somente plugins de disco e CPU. Note que este não é o mesmo que desligar o **tuned-adm**, onde ambos **tuned** e **ktune** estão desabilitados.

***latency-performance***

Um perfil de servidor para um ajuste de desempenho de latência típica. Ele desabilita os mecanismos de economia de energia do **tuned** e **ktune**. O modo **cpuspeed** muda para **performance**. O elevador de E/S foi modificado para **deadline** para cada dispositivo. para

a qualidade do gerenciamento de energia do serviço, o requerimento **cpu_dma_latency**de valor **0** é registrado.

***throughput-performance***

Um perfil de servidor para ajuste de desempenho de rendimento típico. Este perfil é recomendado se o sistema não tiver o armazenamento de classe corporativa. É o mesmo que **latency-performance**, exceto:

- ***kernel.sched_min_granularity_ns*** (granularidade preempção mínima de agendador) é ajustado para **10** milisegundos.
- ***kernel.sched_wakeup_granularity_ns*** (granularidade de ativação de agendador) é ajustado para **15** milisegundos,
- ***vm.dirty_ratio*** (índice de máquina virtual suja) é definido para 40%, e
- transparent huge page são habilitadas.

***enterprise-storage***

Este perfil foi recomendado para configurações de servidores de tamanho corporativo com armazenamento de classe corporativo, incluindo proteção e gerenciamento de cache de controlador com backup de bateria de um cache em disco. É o mesmo que o perfil **throughput-performance**, com uma adição: sistemas de arquivo são re-montadas com o **barrier=0**.

***virtual-guest***

Este perfil é recomendado para configurações de servidor de tamanho corporativo com armazenamento de classe corporativa, incluindo proteção e gerenciamento de cache de controlador com backup de bateria de um cache em disco. É o mesmo que o perfil **desempenho de rendimento**, exceto:

- O valor ***readahead*** é ajustado para **4x**, e
- sistemas de arquivo não root/boot são montados com o **barrier=0**.

***virtual-host***

Baseado no perfil ***enterprise-storage***, o ***virtual-host*** também deminui a troca de memória virtual e habilita um write-back mais agressivo de páginas sujas. Este perfil está disponível no Red Hat Enterprise Linux 6.3 e posteriores, e é o perfil recomendado para os hosts de virtualização, incluindo ambos KVM e hosts Red Hat Enterprise Virtualization.

Consulte o Red Hat Enterprise Linux 6 *Power Management Guide*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, para informações futuras sobre **tuned** e **ktune**.

## 3.5. Perfis de Aplicativo

Profiling é o processo de coleta de informações sobre o comportamento de um programa que é executado. Você analisa um aplicativo para determinar quais áreas de um programa podem ser otimizadas para aumentar a velocidade geral do programa, reduzir o uso de memória, etc. As ferramentas da criação de perfil do aplicativo ajudam a simplificar este processo.

Existem três ferramentas da criação de perfil suportadas para serem usadas com o Red Hat Enterprise Linux 6: **SystemTap** , **OProfile** e **Valgrind** . Documentar essas ferramentas de perfil está fora do escopo deste guia, no entanto, esta seção fornece links para mais informações e uma breve descrição das tarefas para as quais cada criador de perfil é adequado.

### 3.5.1. SystemTap

SystemTap é uma ferramenta de rastreamento e sondagem que permite aos usuários monitorar e analisar as atividades do sistema operacional (particularmente as atividades do kernel) em grande detalhe. Ele fornece informações semelhantes à saída de ferramentas como **netstat**, **top**, **ps** e **iostat** , mas inclui filtragem adicional e opções de análise para as informações que são coletadas.

SystemTap fornece uma análise mais profunda, mais precisa de atividades do sistema e o comportamento do aplicativo para que você possa identificar os afunilamentos do sistema e de aplicativos.

O plugin da função callgraph para o Eclipse usa SystemTap como um back-end, permitindo-lhe controlar completamente o status de um programa, incluindo chamadas de função, retorno, os horários e as variáveis de espaço de usuário, e exibir as informações visualmente para facilitar a otimização.

Para maiores informações sobre o SystemTap consulte o *Guia de Iniciantes do SystemTap*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

### 3.5.2. OProfile

OProfile ( **oprofile** ) é uma ferramenta de monitoramento de desempenho de todo o sistema. Ele usa hardware dedicado de monitoramento de desempenho do processador para obter informações sobre o kernel e do sistema de arquivos executáveis, como quando a memória é referenciada, o número de pedidos do cache L2 e o número de interrupções de hardware recebido. Também pode ser utilizado para determinar a utilização do processador, e as aplicações e serviços que são mais utilizadas.

Oprofile também pode ser usado com o Eclipse via Oprofile Eclipse plug-in. Este plug-in permite aos usuários determinar facilmente as áreas mais demoradas de seu código, e executar todas as funções de linha de comando do OProfile com rica visualização dos resultados.

No entanto, os usuários devem estar cientes de diversas limitações do OProfile:

- As amostras de controle de desempenho podem não ser precisas - porque o processador pode executar instruções fora de ordem, uma amostra pode ser gravada a partir de uma instrução próxima, ao invéz da instrução que gerou a interrupção.
- Como o OProfile é todo o sistema e espera que os processos iniciem e parem várias vezes, amostras de várias execuções são autorizadas a acumular. Isso significa que você pode precisar limpar os dados da amostra de execuções anteriores.
- Ele se concentra na identificação de problemas com os processos de CPU-limitados e, portanto, não identifica os processos que estão inativos enquanto esperam em bloqueios para outros eventos.

Para mais informações sobre como utilizar o OProfile, consulte o *Guia de Implementação*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, ou a documentação **oprofile** em seu sistema, localizada em **/usr/share/doc/oprofile-<*version*>**.

### 3.5.3. Valgrind

Valgrind fornece uma série de ferramentas de detecção e perfil para ajudar a melhorar o desempenho e a correção de suas aplicações. Essas ferramentas podem detectar memória e erros relacionados aos threads, bem como heaps, pilha e saturação de matrizes, o que lhe permite localizar e corrigir erros em seu código de aplicação facilmente. Eles também podem criar perfis no cache, heaps, e ramo de

previsão para identificar os fatores que podem aumentar a velocidade de aplicação e minimizar o uso de memória do aplicativo.

Valgrind analisa o pedido de execução em uma CPU sintético e instrumentar o código do aplicativo existente como ele é executado. Em seguida, imprime "comentário" identificando claramente cada processo envolvido na execução da aplicação de um descritor especificado pelo usuário, arquivo, ou tomada de rede. O nível de instrumentação varia dependendo da ferramenta Valgrind em utilização, e as suas configurações, mas é importante notar que a execução do código instrumentada pode levar 4-50 vezes maior do que a execução normal.

Valgrind pode ser usado em seu aplicativo como está, sem recompilação. No entanto, como o Valgrind utiliza informações de depuração para identificar problemas em seu código, se o seu aplicativo e bibliotecas de apoio não foram compilados com informações de depuração ativada, recompilar para incluir esta informação é altamente recomendado.

Desde o Red Hat Enterprise Linux 6.4, o Valgrind integra com o *gdb* (GNU Project Debugger) para aprimorar eficiência de depuração.

Mais informações sobre o Valgrind estão disponíveis em *Guia do Desenvolvedor*,disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/, ou utilizando o comando **`man valgrind`** quando o pacote *valgrind* for instalado. Documentação extra pode também ser encontrada em:

- **`/usr/share/doc/valgrind-<version>/valgrind_manual.pdf`**
- **`/usr/share/doc/valgrind-<version>/html/index.html`**

Para informaçoes sobre como o Valgrind pode ser utilizado para a memória do sistema, consulte o Seção 5.3, “Utilizando o Valgrind para o Uso de Memória de Perfil ”.

### 3.5.4. Perf

O perf ferramenta fornece um número de contadores de desempenho úteis que permitem ao usuário avaliar o impacto de outros comandos em seu sistema:

**`perf stat`**

Este comando fornece estatísticas globais para eventos de desempenho comuns, incluindo instruções executadas e ciclos de clock consumido. Você pode usar os sinalizadores de opção para reunir estatísticas sobre outros do que os eventos de medição padrão de eventos. A partir do Red Hat Enterprise Linux 6.4, é possível usar **`perf status`** para filtrar monitoramento baseado em um ou mais grupos de controle especificados (cgroups). Para mais informações, leia a página man: **`homem perf-stat`** .

**`perf record`**

This command records performance data into a file which can be later analyzed using **`perf report`**. For further details, read the man page: **`man perf-record`**.

**`perf report`**

This command reads the performance data from a file and analyzes the recorded data. For further details, read the man page: **`man perf-report`**.

**`perf list`**

This command lists the events available on a particular machine. These events will vary based

on the performance monitoring hardware and the software configuration of the system. For further information, read the man page: **`man perf-list`**.

**`perf top`**

This command performs a similar function to the **`top`** tool. It generates and displays a performance counter profile in realtime. For further information, read the man page: **`man perf-top`**.

Mais informações sobre o **`perf`** estão disponíveis no Red Hat Enterprise Linux *Developer Guide*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

## 3.6. Red Hat Enterprise MRG

Componente Realtime do Red Hat Enterprise MRG inclui **`Tuna`** , uma ferramenta que permite aos usuários ajustar os valores ajustáveis de seus sistemas e ver os resultados dessas mudanças. Embora tenha sido desenvolvido para utilização com o componente em tempo real, ele também pode ser usado para sintonizar sistemas padrão do Red Hat Enterprise Linux.

Com o Tuna, você pode ajustar ou desabilitar atividades de sistemas desnecessários, incluindo:

- Parâmetros BIOS relacionados ao gerenciamento de energia, detecção de erros e interrupções de gerenciamento de sistema;
- configurações de rede, tais como união de interrupção e o uso do TCP;
- atividade de agendamento nos sistemas de arquivo de agendamento;
- autenticação de sistema;
- se a interrupção e processos de usuário são manuseados por uma CPU específica ou uma classe de CPUs;
- se o espaço swap é utilizado; e
- como lidar com exceções sem memória

Para obter informações conceituais mais detalhadas sobre o ajuste do Red Hat Enterprise MRG com a interface Tuna, consulte o "Sintonia Geral do Sistema" capítulo do *Guia de Ajuste Realtime*. Para obter instruções detalhadas sobre como usar a interface Tuna, consulte o *Guia do Usuário Tuna*. Ambas as guias estão disponíveis http://access.redhat.com/site/documentation/Red_Hat_Enterprise_MRG/.

# Capítulo 4. CPU

O termo CPU, o que significa *unidade central de processamento* , é um equívoco para a maioria dos sistemas, uma vez que *centro* implica *único*, enquanto que a maioria dos sistemas modernos têm mais de uma unidade de processamento, ou o núcleo. Fisicamente, as CPUs estão contidas num pacote ligado a uma placa-mãe em um *soquete*. Cada soquete na placa-mãe tem várias conexões:com outros soquetes da CPU, controladores de memória, controladores de interrupção e outros dispositivos periféricos. O soquete para o sistema operacional é um agrupamento lógico de recursos de CPUs e associados. Este conceito é central para a maioria das nossas discussões sobre o ajuste da CPU.

O Red Hat Enterprise Linux mantém uma riqueza de estatísticas sobre os eventos de CPU do sistema, estas estatísticas são úteis no planejamento de uma estratégia de ajuste para melhorar o desempenho da CPU. Seção 4.1.2, "Ajustando Desempenho de CPU" discute algumas das estatísticas mais úteis, onde encontrá-las, e como analisá-las para ajuste de desempenho.

## Topologia

Computadores mais antigos tinham relativamente poucas CPUs por sistema, o que permitia uma arquitetura conhecida como *Symmetric Multi-Processor* (SMP). Isto significa que cada CPU no sistema semelhante (ou simétrico) tinha acesso à memória disponível. Nos últimos anos, a contagem de CPU por soquete cresceu até ao ponto que a tentativa de dar acesso a toda a memória RAM simétrica no sistema tornou-se muito dispendiosa. Mais elevados sistemas de contagem de CPU estes dias possuem uma arquitetura conhecida como *Non-Uniform Memory Access* (NUMA) ao invés de SMP.

Os processadores AMD tinham este tipo de arquitetura durante algum tempo com suas interconexões *Hyper Transport* (HT), enquanto a Intel começou a implementar NUMA em suas criações de *Interconexão de Caminho Rápida* (QPI). O NUMA e SMP são ajustados de forma diferente, pois você precisa levar em conta a *topologia* do sistema ao alocar recursos para um aplicativo.

## Threads

Dentro do sistema operacional Linux, a unidade de execução é conhecida como um *thread* . Threads tem um contexto de registro, uma pilha e um segmento de código executável, o qual eles executam em uma CPU. É o trabalho do sistema operacional (OS) de agendar esses tópicos sobre as CPUs disponíveis.

O OS maximiza a utilização da CPU de balanceamento de carga dos threads por núcleos disponíveis. Uma vez que o sistema operacional é principalmente preocupado com a manutenção de CPUs ocupados, ele pode não tomar decisões adequadas com relação ao desempenho do aplicativo. Mover um segmento do aplicativo para a CPU em outro soquete pode piorar o desempenho mais do que simplesmente esperar a CPU atual se tornar disponível, uma vez que as operações de acesso à memória podem diminuir drasticamente através de sockets. Para aplicações de alto desempenho, geralmente é melhor o designer determinar onde os tópicos devem ser colocados. O Seção 4.2, "Agendamento da CPU" discute a melhor forma de alocar a CPU e memória para melhor executar threads da aplicação.

## Interrupções

Um dos eventos de sistema menos óbvios (mas importante) que podem afetar o desempenho do aplicativo é a *interrupção* (também conhecida como IRQs em Linux). Estes eventos são tratados pelo sistema operacional, e são utilizados pelos periféricos para sinalizar a chegada de dados ou a conclusão de uma operação, como a gravação de rede ou um evento timer.

A maneira pela qual o sistema operacional ou processador que está executando o código do aplicativo

lida com uma interrupção não afeta a funcionalidade do aplicativo. No entanto, pode afetar o desempenho da aplicação. Este capítulo também discute dicas sobre prevenção de interrupções que impactam negativamente o desempenho do aplicativo.

## 4.1. Topologia da CPU

### 4.1.1. CPU e a Topologia NUMA

Os primeiros processadores de computador foram *uniprocessadores* , o que significa que o sistema tinha uma única CPU. A ilusão de executar processos em paralelo foi feita pelo sistema operacional mudando rapidamente o único CPU de um thread de execução (processo) para outro. Na procura de aumentar o desempenho do sistema, os criadores notaram que o aumento da taxa de relógio para executar instruções mais rápido funcionava apenas até um ponto (geralmente as limitações ao criar uma forma de onda de relógio estável com a tecnologia actual). Em um esforço para obter um desempenho mais geral do sistema, os criadores acrescentaram outra CPU para o sistema, permitindo duas correntes paralelas de execução. Esta tendência de adicionar processadores tem continuado ao longo do tempo.

Sistemas multiprocessadores mais antigos foram projetados de modo que cada CPU tinha o mesmo caminho lógico para cada local de memória (geralmente um barramento paralelo). Isto deixa cada CPU acessar qualquer local de memória na mesma quantidade de tempo, como qualquer outra CPU no sistema. Esse tipo de arquitetura é conhecida como um sistema multi-processador (SMP) simétrico. SMP é bom para algumas CPUs, mas uma vez que a contagem de CPU fica acima de um certo ponto (8 ou 16), o número de rastreamentos paralelos necessários para permitir a igualdade de acesso à memória, usa muito do estado real da placa disponível, deixando menos espaço para periféricos.

Dois novos conceitos combinados para permitir um número maior de CPUs em um sistema:

1. Barramentos em série
2. Topologias NUMA

Um barramento de série é um caminho de comunicação de um único fio com uma taxa muito elevada de relógio, que transfere dados como intermitências em pacotes. Criadores de hardware começaram a usar barramentos seriais como interconexões de alta velocidade entre as CPUs, e entre os processadores e controladores de memória e outros periféricos. Isto significa que, em vez de exigir entre 32 e 64 rastreamentos na placa de *cada* CPU para o subsistema de memória, agora existe *um* rastreamento, reduzindo substancialmente a quantidade de espaço necessária na placa.

Ao mesmo tempo, os criadores de hardware empacotavam mais transistores no mesmo espaço reduzindo tamanhos de matrizes. Em vez de colocar CPUs individuais diretamente na placa principal, eles começaram a empacotá-las em um pacote de processador como processadores multi-core. Então, em vez de tentar proporcionar igualdade de acesso à memória de cada pacote do processador, os criadores recorreram a estratégia de acesso de memória não-uniforme (NUMA), onde cada combinação de pacote / socket possui um ou mais área de memória dedicada para acesso de alta velocidade. Cada socket também possui uma interconexão com outras bases de acesso mais lento à memória dos outros sockets.

Como um exemplo simples do NUMA, suponha que temos uma placa-mãe de dois soquetes, onde cada soquete foi preenchido com um pacote de quad-core. Isso significa que o número total de processadores no sistema é de oito; quatro em cada tomada. Cada tomada também tem um banco de memória anexado a quatro gigabytes de RAM, para uma memória total do sistema de oito gigabytes. Como propósito deste exemplo, processadores 0-3 estão em soquete 0, e as CPUs 4-7 estão no socket 1. Cada socket neste exemplo também corresponde a um nó NUMA.

Pode levar três ciclos de relógio para que a CPU 0 acesse a memória do banco 0: um ciclo para apresentar o endereço ao controlador de memória, um ciclo para configurar o acesso no local de memória, e um ciclo para ler ou gravar no local. No entanto, pode levar seis ciclos de relógio para a CPU 4 acessar a memória do mesmo local, pois como está em um soquete separado, ele deve passar por dois controladores de memória: o controlador de memória local no socket 1, e, em seguida, o controlador de memória remoto no soquete 0. Se a memória é contestada naquele local (ou seja, se mais de uma CPU estiver tentando acessar o mesmo lugar ao mesmo tempo), controladores de memória precisarão arbitrar e serializar o acesso à memória, por isso o acesso à memória demorará mais tempo. Adicionando consistência de cache (garantindo que caches da CPU locais contenham os mesmos dados para o mesmo local da memória) complica ainda mais o processo.

Os processadores de alta tecnologia mais recentes, tanto da Intel (Xeon) quanto da AMD (Opteron), possuiem topologias NUMA. Os processadores AMD utilizam uma interconexão conhecida como HyperTransport, ou HT, enquanto a Intel usa um chamado QuickPath Interconnect, ou QPI. As interconexões diferem na forma como se conectam fisicamente com outras interconexões, memória ou dispositivos periféricos, mas na verdade eles são a chave que permite o acesso transparente para um dispositivo conectado a partir de outro dispositivo conectado. Neste caso, o termo 'transparente' se refere ao facto de que não há nenhuma API de programação especial necessária para utilizar a interconexão, não a opção "sem custo".

Como as arquiteturas de sistema são tão diversas, é impraticável caracterizar especificamente a penalidade de desempenho imposto ao acessar a memória não-local. Podemos dizer que cada *hop* em uma interconexão impõe pelo menos alguma perda de desempenho relativamente constante por hop, assim referenciando uma posição de memória que seja duas interconexões a partir da CPU atual, impõe pelo menos *2N + tempo de ciclo de memória* unidades para acessar tempo, onde N é a penalidade por hop.

Dada essa penalidade de desempenho, os aplicativos de desempenho sensíveis devem evitar acessar regularmente a memória remota em um sistema de topologia NUMA. O aplicativo deve ser configurado para que ele permaneça em um nó privado e que aloque memória daquele nó.

Para fazer isto, existem algumas coisas que este aplicativo precisará saber:

1. Qual a *topologia* do sistema?
2. Onde o aplicativo está executando atualmente?
3. Onde está o banco de memória mais próximo?

### 4.1.2. Ajustando Desempenho de CPU

Leia esta seção para entender como ajustar para obter melhor desempenho de CPU,

NUMA foi originalmente usado para conectar um único processador com vários bancos de memória. A medida que os fabricantes de CPU refinaram seus processos e os tamanhos de dados encolheram, múltiplos núcleos de CPU puderam ser incluídos em um pacote. Esses núcleos foram agrupados de modo que cada um deles tinha tempo de acesso igual a um banco de memória local, e o cache pôde ser compartilhado entre os núcleos, no entanto, cada 'hop' em uma interligação entre o núcleo, memória, e cache envolve uma pequena perda de desempenho.

O sistema de exemplo em Figura 4.1, “Acesso Remoto e Local de Memória em topologia NUMA” contém dois nós NUMA. Cada nó tem quatro processadores, um banco de memória, e um controlador de memória. Qualquer CPU em um nó tem acesso direto ao banco de memória nesse nó. Seguindo as setas no nó 1, os passos são como se segue:

1. Uma CPU (qualquer um dos 0-3) apresenta o endereço de memória para o controlador da memória local.

2. O controlador de memória define o acesso ao endereço de memória.
3. A CPU executa a leitura ou gravação de operações naquele endereço de memória.

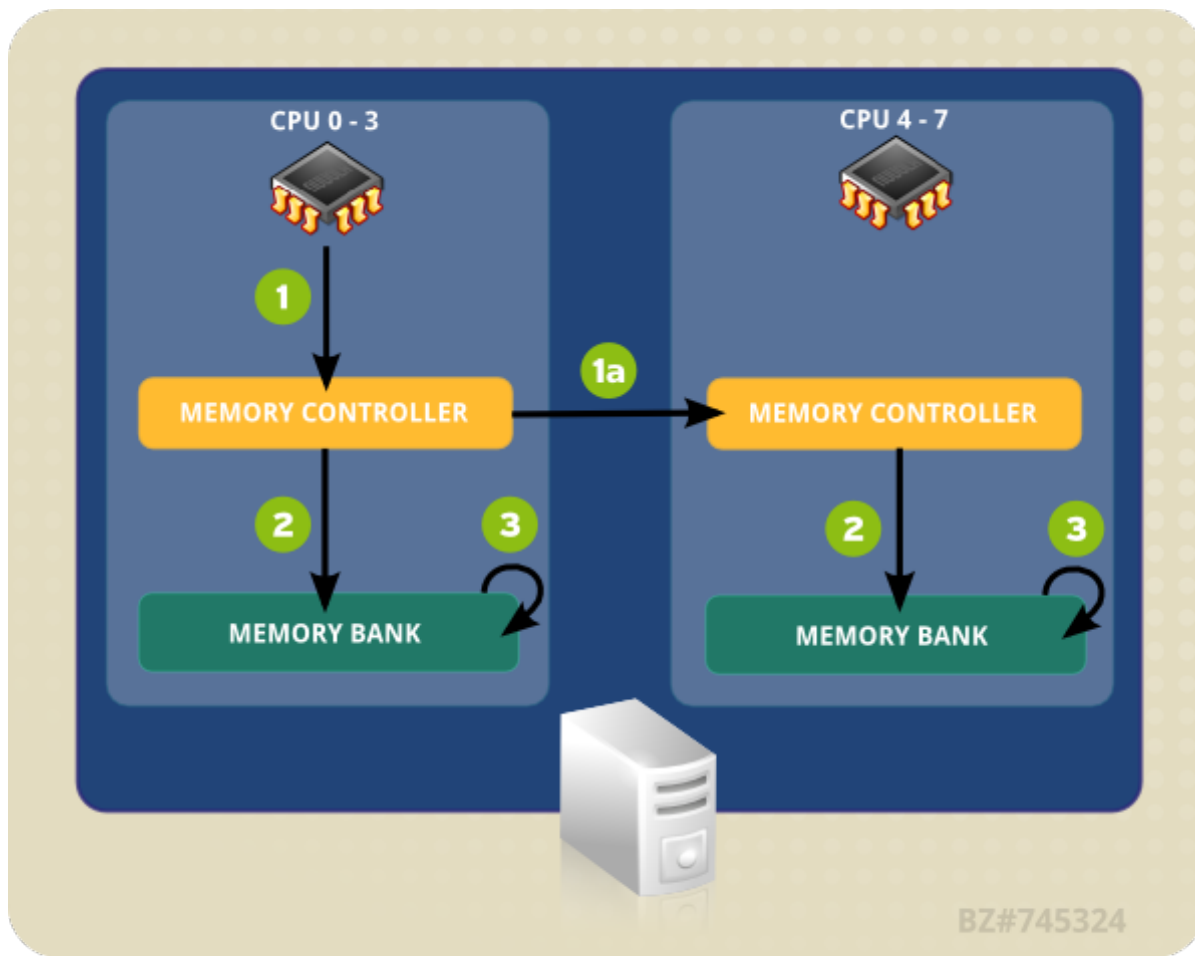



**Figura 4.1. Acesso Remoto e Local de Memória em topologia NUMA**

No entanto, se uma CPU em um nó precisa acessar o código que reside no banco de memória de um nó NUMA diferente, o caminho que deve tomar é menos direto:

1. Uma CPU (qualquer um dos 0-3) apresenta o endereço de memória remota ao controlador da memória local.
   a. A requisição da CPU para o endereço de memória remota é passado para um controlador de memória remota, local para o nó que contém o endereço de memória.
2. O controlador de memória remota define o acesso ao endereço de memória remota.
3. A CPU executa a leitura ou gravação de operações naquele endereço de memória remota.

Toda ação precisa passar por vários controladores de memória, para que o acesso possa demorar mais do que o dobro do tempo ao tentar acessar endereços de memória remota. A principal preocupação com o desempenho em um sistema de vários núcleos é, portanto, garantir que a informação viaje tão eficientemente quanto possível, através do mais curto, ou mais rápido, caminho.

Para configurar um aplicativo para o melhor desempenho da CPU, você precisa saber:

- a topologia do sistema (como os seus componentes estão conectados),
- o núcleo no qual o aplicativo é executado, e

- a localização do banco de memória mais próximo.

O Red Hat Enterprise Linux 6 é apresentado com uma série de ferramentas para ajudá-lo a encontrar essas informações e ajustar o seu sistema de acordo com suas necessidades. As seções a seguir fornecem uma visão geral das ferramentas úteis para ajuste do desempenho da CPU.

#### 4.1.2.1. Configuração, Afinidade da CPU com o taskset

O **taskset** recupera e define a afinidade de CPU de um processo em execução (por processo de ID). Ele também pode ser usado para iniciar um processo com uma determinada afinidade da CPU, que vincula o processo especificado à uma CPU especificada ou conjunto de CPUs. No entanto, **taskset** não vai garantir a alocação de memória local. Se você precisar dos benefícios de desempenho adicionais de alocação de memória local, recomendamos **numactl** sobre **taskset** , veja Seção 4.1.2.2, "Controlling NUMA Policy with **numactl**" para mais detalhes.

A CPU afinidade é representado como um bitmask. O bit de ordem mais baixa corresponde à primeira CPU lógica, e o bit de ordem mais alta corresponde à última CPU lógica. Estas máscaras são normalmente fornecidas em hexadecimal, assim o **0x00000001** representa o processador 0, e o **0x00000003** representa processadores 0 e 1.

Para definir a afinidade da CPU de um processo em execução, execute o seguinte comando, substituindo ***mask*** pela máscara do processador ou processadores que você quer que o processo se vincule, e ***pid*** pelo ID do processo do processo cuja afinidade que você deseja alterar.

```
# taskset -p mask pid
```

Para iniciar um processo com uma determinada afinidade, execute o seguinte comando, substituindo ***mask*** pela máscara do processador ou processadores que você quer que o processo se vincule, e ***program*** pelo programa , opções e argumentos do programa que você deseja executar.

```
# taskset mask -- program
```

Em vez de especificar os processadores como uma máscara de bits, você também pode usar o **-c** opção para fornecer uma lista delimitada por vírgulas de processadores separados, ou uma variedade de processadores, assim:

```
# taskset -c 0,5,7-9 -- myprogram
```

Outras informações sobre **taskset** está disponível na página de man: **homem taskset** .

#### 4.1.2.2. Controlling NUMA Policy with numactl

**numactl** executa processos com um agendamento específico ou política de colocação de memória. A política selecionado está definido para o processo e todos os seus filhos. **numactl** pode também definir uma política persistente de segmentos de memória compartilhada ou arquivos e definir a afinidade da CPU e afinidade de memória de um processo. Ele usa o **/ sys** sistema de arquivos para determinar a topologia do sistema.

O **/ sys** sistema de arquivos contém informações sobre como CPUs, memória e dispositivos periféricos são conectados via NUMA interconexões. Especificamente, o **/ sys / devices /system/cpu** contém informações sobre como CPUs de um sistema estão ligados um ao outro. O **/ sys / devices / system / node** contém informações sobre os nós NUMA no sistema, e as distâncias relativas entre os nós.

Num sistema NUMA, quanto maior a distância entre um processador e um banco de memória, mais lento

o acesso do processador para que banco de memória. Aplicações sensíveis ao desempenho deve ser configurado de forma que eles alocar memória o mais próximo possível do banco de memória.

Desempenho aplicações sensíveis devem também ser configurado para executar em um determinado número de núcleos, particularmente no caso de aplicações multi-threaded. Porque caches de primeiro nível são geralmente pequenas, se vários segmentos executar em um núcleo, cada segmento potencialmente expulsar os dados em cache acessados por um fio anterior. Quando o sistema operacional tenta multitarefa entre estes tópicos, e os fios continuam a despejar uns dos outros dados em cache, uma grande porcentagem de seu tempo de execução é gasto em substituição linha de cache. Este problema é conhecido como *cache de goleada* . Portanto, recomenda-se ligar uma aplicação multi-threaded para um nó ao invés de um único núcleo, uma vez que este permite que os fios para compartilhar linhas de cache em vários níveis (cache de primeira,, segunda e último nível) e minimiza a necessidade de armazenar em cache encher operações. No entanto, a ligação de um aplicativo para um único núcleo pode ser performance se os tópicos estão acessando os mesmos dados em cache.

**numactl** permite vincular um aplicativo para um núcleo específico ou nó NUMA, e alocar a memória associada a um núcleo ou conjunto de núcleos para esse aplicativo. Algumas opções úteis fornecidas pelos numactl são:

***--show***

Mostrar as definições de política NUMA do processo atual. Este parâmetro não necessita de outros parâmetros, e pode ser usado assim: **numactl - espetáculo** .

***--hardware***

Exibe um inventário de nós disponíveis no sistema

***--membind***

Somente alocar memória dos nós especificados. Quando esta estiver em uso, a alocação falhará se a memória nesses nós é insuficiente. Uso para este parâmetro é **numactl - membind = *nós* *programa*** , onde ***nós*** é a lista de nós que você quer para alocar memória e ***programa*** é o programa cujos requisitos de memória deve ser alocado a partir desse nó. Números nó pode ser dada como uma lista delimitado por vírgulas, um intervalo, ou uma combinação dos dois. Mais detalhes estão disponíveis no **numactl** página man: **homem numactl** .

***--cpunodebind***

Só executar um comando (e seus processos filhos) em CPUs que pertencem ao nó especificado (s). Uso para este parâmetro é **numactl - cpunodebind = *nós* *programa*** , onde ***nós*** é a lista de nós a cuja CPUs o programa especificado ( ***programa*** ) deve estar vinculado. Números nó pode ser dada como uma lista delimitado por vírgulas, um intervalo, ou uma combinação dos dois. Mais detalhes estão disponíveis no **numactl** página man: **man numactl** .

***--physcpubind***

Somente execute um comando (e seus processos filho) em CPUs especificadas. O uso para este parâmetro é **numactl --physcpubind=*cpu* *program***, onde ***cpu*** é uma lista separada por vírgulas de números de CPU físicos como exibido nos campos do processador do**/proc/cpuinfo**, e ***program*** é o programaque deve executar somente naqueles CPUs. Os CPUs podem também ser especificados dependendo do **cpuset** atual. Consulte a página man do **numactl** para obter mais informações: **man numactl**.

***--localalloc***
: Especifica se a memória deve sempre ser alocada no nó atual.

***--preferred***
: Sempre que possível, a memória é alocada no nó especificado. Se a memória não pode ser alocada no nó especificado, cair de volta para os outros nós. Esta opção tem apenas um único número de nó, assim: **numactl - preferred =** ***nó*** . Consulte a página man numactl para mais informações: **homem numactl** .

O **libnuma** biblioteca incluída no *numactl* pacote oferece uma interface de programação simples para a política NUMA suportado pelo kernel. É útil para mais ajuste refinado do que o **numactl** utilidade. Mais informações estão disponíveis na página do homem: o homem **NUMA (3)** .

### 4.1.3. numastat

**Importante**

Anteriormente, a ferramenta **numastat** era um script do Perl escrito por Andi Kleen. Ele foi reescrito de forma significante para o Red Hat Enterprise Linux 6.4.
Embora o comando padrão (**numastat**, sem nenhuma opção ou parâmetro) mantém a compatibilidade severa com as versões anteriores da ferramenta, note que as opções ou parâmetros fornecidos à este comando, muda significantemente o conteúdo de resultado e seu formato.

O **numastat** exibe a estatística de memória (tal como as alocações de acertos e erros) para processos e o sistema operacional baseado em nó de NUMA. Por padrão, executar o **numastat** exibe como muitas páginas de memória estão ocupadas pelas seguintes categorias de evento para cada nó.

O desempenho da CPU adequado é indicado por baixo **numa_miss** e valores **numa_foreign**.

Esta versão atualizada do **numastat** também mostra se a memória do processo é distribuiída em um sistema ou centralizada em nós específicos utilizando o **numactl**.

O resultado de referência cruzada do**numastat** com o resultado **top** por CPU, para verificar se os threads do processo estão executando no mesmos nós para o qual a memória foi alocada.

**Categorias de Rastreamento Padrão**

**numa_hit**
: O número de tentativas de alocações neste nós que foram bem sucedidas.

**numa_miss**
: O número de tentativas de alocações em outro nó que foram alocadas neste nó porcausa da baixa memória no nó pretendido. Cada evento **numa_miss** possui um evento **numa_foreign** correspondente em outro nó.

**numa_foreign**

O número de alocações pretendidas inicialmente para este nó que foram alocadas à outro nó. Cada evento **numa_foreign** possui um evento **numa_miss** correspondente em outro nó.

**interleave_hit**

O número de tentativas de alocações de políticas de intercalação neste nó que foram bem sucedidas.

**local_node**

O número de vezes que um processo neste nó alocou memória com sucesso neste nó.

**other_node**

O número de vezes que um processo em outro nó alocou memória neste nó.

Fornecer qualquer outra mudança muda as unidades exibidas em megabytes de memória (arredondadas para dois decimais) e modifica outros comportamentos específicos do **numastat**como descrito abaixo.

**-c**

Horizontalmente condensa a tabela da informação exibida. Isso é útil em sistemas com um elevado número de nós NUMA, mas a largura da coluna e do espaçamento entre colunas são um tanto imprevisíveis. Quando esta opção é utilizada, a quantidade de memória é arredondado para o próximo megabyte.

**-m**

Exibe as informações de uso de memória em todo o sistema baseado por nó, semelhante à informação encontrada em **/proc/meminfo**.

**-n**

Exibe a mesma informação que o comando original **numastat** (numa_hit, numa_miss, numa_foreign, interleave_hit, local_node, e other_node), com um formato atualizado, utilizando megabytes como unidade de medida.

**-p *pattern***

Exibe informações por nó para o modelo específico. Se o valor para o ***modelo*** é composto de dígitos, o **numastat** entende que seja um identificador de processo numérico. Caso contrário, o **numastat** procura por linhas de comando do processo para um modelo em específico.

Os argumentos de linha de comando inseridos após o valor da opção **-p** devem ser modelos adicionais para o qual filtrar. Modelos adicionais expandem, ao em vez de estreitarem o filtro.

**-s**

Filtra os dados exibidos em ordem descendente para que os maiores consumidores de memória (de acordo com a coluna **total**) são listados primeiro.

Opcionalmente, você pode especificar o ***node***, e a tabela será filtrada de acordo com a coluna do ***node***.Ao utilizar esta opção, o valor do ***node***deve seguir a opção **-s** imediatamente, como

demonstrado aqui:

```
numastat -s2
```

Não inclui o espaço em branco entre a opção e seu valor.

**-v**

Exibe mais informações de verbosidade. De modo que as informações do processo para processos múltiplos exibirão informações detalhadas para cada processo.

**-V**

Exibe as informações da versão **numastat**.

**-z**

Omite a faixa da tabela e colunas com valores zero de informações exibidas. Note que alguns valores quase zero que são arredondados para zero para exibir propósitos, não serão omitidos do resultado exibido.

## 4.1.4. NUMA Daemon de Gerenciamento de Afinidade (numad)

**numad**é um daemon de gerenciamento de afinidade do NUMA automático. Ele monitora a topologia do e uso de recursos dentro de um sistema para aprimorar a alocação e gerenciamento de recurso do NUMA de forma dinâmica (e assim o desempenho do sistema).

Dependendo da carga de trabalho do sistema, **numad** pode fornecer as melhorias de desempenho do parâmetro de comparação do **numad** acessa informações periodicamente a partir do sistema de arquivo do **/proc** para monitorar recursos de sistemas disponíveis por nó. O daemon então tenta colocar processos significativos em nós do NUMA que possuam memória alinhada suficientes e recursos de CPU para o desempenho adequado do NUMA. Limites atuais para o gerenciamento do processo são de ao menos 50% de uma CPU e de ao menos 300 MB de memória. O **numad** tenta manter um nível de uso de recurso e rebalancear alocações quando necessárias, movendo processos entre nós de NUMA.

O **numad** também fornece um serviço de conselho de pré-colocação que pode ser pesquisado por diversos sistemas de gerenciamento de empregos para fornecer assistência com a vinculação inicial de recursos da CPU e memória para seus processos. Este serviço de conselho de pré-colocação está disponível não importando se o **numad** está executando como um daemon em um sistema. Consulte as páginas man para detalhes futuros sobre a utilização da opção **-w** para o aconselhamento de pré-colocação: **man numad**.

### 4.1.4.1. Benefícios do numad

O **numad**beneficia primeiramente os sistemas com processos de longa duração que consomem quantidades significativas de recursos, especialmente quando esses processos estão contidos em um subconjunto do total de recursos do sistema.

**numad** também pode beneficiar aplicativos que consomem nós NUMA que valem os recursos. No entanto, os benefícios que o **numad** fornece, diminui a medida que a porcentagem de recursos consumidos em um sistema aumenta.

É improvável que o **numad** melhore o desempenho quando os processos são executados por apenas

alguns minutos, ou não consomem muitos recursos. Sistemas com padrões contínuos imprevisíveis memória de acesso, como grandes bancos de dados na memória, são também susceptíveis de beneficiar do uso do **NUMAD** .

#### 4.1.4.2. Modos de operação

**Nota**

Estatísticas de contabilidade de memória do kernel podem se contradizer depois de grandes quantidades de fusão. Como tal, o **NUMAD** pode ser confuso quando o daemon KSM mescla grandes quantidades de memória. O daemon KSM será mais focado em NUMA em versões futuras. No entanto, atualmente, se o seu sistema tem uma grande quantidade de memória livre, você pode atingir um melhor desempenho, desligando e desabilitando o daemon KSM.

**numad** pode ser utilizado de duas formas:

- como um serviço
- como um executável

##### 4.1.4.2.1. Utilizando o numad como um serviço

Enquanto o serviço **numad** é executado, ele tentará ajustar de forma dinâmica o sistema baseado em sua carga de trabalho.

Para iniciar um serviço, execute:

```
# service numad start
```

Para fazer com que o serviço persista em reinicializações, execute:

```
# chkconfig numad on
```

##### 4.1.4.2.2. Usando o numadcomo um executável

Para usar o **numad** como um executável, execute somente o:

```
# numad
```

**numad** será executado até que seja interrompido. Enquanto estiver em execução, suas atividades serão autenticadas em **/var/log/numad.log**.

Para restringir o gerenciamento do **numad** à um processo específico, inicie-o com as seguintes opções.

```
# numad -S 0 -p pid
```

***-p pid***

Adiciona o ***pid*** especificado à uma lista de inclusão explícita. O processo especificado não será gerenciado até que ele atenda ao limite de significância do processo do **numad**.

***-S mode***

O parâmetro do ***-S*** especifica o tipo de escaneamento do processo. Configurá-lo para **0** como demonstrado, limita o gerenciamento do **numad** para processos explicitamente incluídos.

Para interromper o **numad**, execute:

```
# numad -i 0
```

Interromper o **numad** não remove as mudanças que fez para aprimorar a afinidade do NUMA. Se o sistema utilizar mudanças de forma significativa, a execução do **numad** novamente irá ajustar a afinidade para aprimorar o desempenho sob as novas condições.

Para informações futuras sobre as opções do **numad** disponíveis, consulte a página man **numad**: **`man numad`**.

## 4.2. Agendamento da CPU

O *agendador* é responsável em manter as CPUs ocupadas no sistema. O agendador do Linux implementa um número de *políticas de agendamento*, que determina quando e por quanto tempo uma thread foi executada em um núcleo da CPU em específico.

As políticas de agendamento são divididas em duas categorias principais:

1. Políticas em Tempo Real (Realtime)
   - SCHED_FIFO
   - SCHED_RR
2. Políticas Normais
   - SCHED_OTHER
   - SCHED_BATCH
   - SCHED_IDLE

### 4.2.1. Políticas de agendamento em Tempo Real (Realtime)

As threads em tempo real são agendadas primeiro, e threads normais são agendadas depois que todas as threads em tempo real serem agendadas.

As políticas *tempo real* são utilizadas para tarefas de horário crítico que devem ser concluídas sem interrupções.

**`SCHED_FIFO`**

Esta política também é mencionada como *agendamento de prioridade estática*, pois ela define uma prioridade fixa (entre 1 e 99) para cada thread. O agendador copia uma lista de threads SCHED_FIFO para o thread de maior prioridade que esteja pronto para ser executado. Esta thread é executada até que seja bloqueada, ou tenha admitido preempção por um thread de prioridade maior que esteja pronto para ser executado.

Até mesmo a thread em tempo real com a prioridade mais baixa será agendada antes do que qualquer thread com uma política não-tempo real; Se somente existir threads em tempo real, o valor de prioridade **`SCHED_FIFO`** não importará.

**`SCHED_RR`**

Uma variante de repetição alternada (round-robin) da política **`SCHED_FIFO`**. As threads do **`SCHED_RR`** também recebem uma prioridade fixa entre 1 e 99. No entanto, as threads com a mesma prioridade são agendadas em estilo repetição alternada dentro de um certo quantum,

ou fração de tempo. A chamada do sistema **sched_rr_get_interval(2)** retorna o valor de fração de tempo, mas a duração da fração de tempo não pode ser estabelecida por um usuário. Esta política é útil se você precisa executar threads múltiplas na mesma prioridade.

Para mais informações detalhadas sobre a semântica definida das políticas de agendamento em tempo real, consulte o *IEEE 1003.1 POSIX standard* Sob Interfaces de Sistema — Realtime, a qual está disponível em http://pubs.opengroup.org/onlinepubs/009695399/functions/xsh_chap02_08.html.

A melhor forma de definir a prioridade de threads é começar com uma prioridade baixa e aumentar quando uma latência legitima for identificada. As threads Realtime threads nãopossuem o tempo fraccionado como threads normais; As threads **SCHED_FIFO** funcionam até que sejam bloqueadas, retiradas ou pré-esvaziadas por uma thread com uma prioridade maior. Configurar uma prioridade de 99 é portanto desencorajado, pois isto colocaria seus processos no mesmo nível de prioridade que a thread de migração e as threads watchdog. Se estas threads forem bloqueadas porque sua thread entrou no loop computacional, estes não serão executados. Os sistemas de uniprocessador será bloqueado neste caso.

No kernel do Linux, a política **SCHED_FIFO** inclui o mecanismo do pacote de largura de banda. Isto projeta programadores de aplicativos de realtime de tarefas realtime que podem monopolizar a CPU. Este mecanismo pode ser ajustado através dos seguintes parâmetros de sistema de arquivo **/proc** :

**/proc/sys/kernel/sched_rt_period_us**

Define o período de tempo a ser considerado cem porcento da largura de banda de CPU, em microsegundos ('us' sendo o texto simples mais próximo de 'µs'). O valor parão é de 1000000µs, ou 1 segundo.

**/proc/sys/kernel/sched_rt_runtime_us**

Define o período de tempo a ser devotado a threads de realtime em execução, em microsegundos ('us' sendo o texto simples mais próximo de 'µs'). O valor parão é de 950000µs, ou 0.95 segundos.

### 4.2.2. Políticas de agendamento normal

Existem três políticas de agendamento normais: **SCHED_OTHER**, **SCHED_BATCH** e **SCHED_IDLE**. No entanto, as políticas **SCHED_BATCH** e **SCHED_IDLE** pretendem ser direcionadas para trabalhos com baixa prioridade, e como tal, são de interesse limitado em um guia de ajuste de desempenho.

**SCHED_OTHER, ou SCHED_NORMAL**

A política de agendamento padrão. Esta polícia utiliza o Agendador Totalmente Justo (Completely Fair Scheduler - CFS) para fornecer períodos de acesso justos para todas as threads utilizando esta política. Os CFS estabelecem uma lista de prioridades dinâmicas baseadas em partes no valor ***niceness*** de cada thread do processo. (Consulte o *Guia de Implementação* para mais detalhes sobre este parâmetro e o sistema de arquivo **/proc**.) Isto fornece aos usuários um nível indireto de controle sob a prioridade de processo, mas a lista de prioridade dinâmica pode somente ser modificada diretamente pelo CFS.

### 4.2.3. Seleção da política

Selecionar a política do agendador correta para threads de uma aplicação nem sempre é uma tarefa

simples. Em geral, as políticas em tempo real deve ser usada para o tempo de tarefas críticas ou importantes que têm de ser agendados de forma rápida e não serem executados por períodos de tempo prolongados. Políticas normais geralmente trazem melhores resultados do que as políticas de transferência de dados em tempo real porque eles permitem que o agendador execute segmentos de forma mais eficiente (ou seja, eles não precisam reagendar para a preempção como sempre).

Se você estiver gerenciando um grande número de threads e estiver preocupado principalmente com o rendimento de dados (os pacotes de rede por segundo, gravações de disco, etc) então utilize o **SCHED_OTHER** e deixe o sistema gerenciar o uso da CPU para você.

Se você estiver preocupado com o tempo de resposta do evento (latência), use o **SCHED_FIFO**. Caso você tenha um número de threads pequeno, considere isolar um soquete de CPU e mover suas threads para aqueles núcleos de soquete, assim não existirão outras threads competindo por tempo nos núcleos.

## 4.3. Interrupções e Ajuste de IRQ

Uma requisição de interrupção (IRQ) é uma requisição para serviço, enviada em nível de hardware. Interrupções podem ser enviadas por linhas de hardware dedicadas ou através de um bus de hardware como um pacote de informações(um Message Signaled Interrupt, ou MSI).

Quando as interrupções são habilitadas, o recebimento de um IRQ solicita uma troca para contexto de interrupção. O código de expedição de interrupções do Kernel recupera o número do IRQ e sua lista de associados de Rotinas de Serviço de Interrupçao (ISRs) registradas, e chama cada ISR. O ISR é consciente das interruções e ignora interrupções redundantes do mesmo IRQ, depois enfileira um manuseador deferido para terminar o processamento de interrupções e evitar que o ISR ignore interrupções futuras.

O arquivo **/proc/interrupts** relaciona o número de interrupções por CPU por dispositivo de E/S. Ele exibe o número de IRQ, o número daquela interrupção manipulada por cada núcleo da CPU, o tipo de interrupção, e uma lista delimitada por vírgulas de motoristas que estão registrados para receber essa interrupção. (Consulte a página de manual proc (5) para mais detalhes: **man 5 proc**)

Os IRQs possuem uma propriedade de "afinidade" associada, ***smp_affinity***, que define os núcleos da CPU que podem executar o ISR para aquele IRQ. Esta propriedade pode ser usada para aprimorar o desempenho de aplicativo atribuindo a afinidade de interrupção e a afinidade do thread do aplicativo para um ou mais núcleos específicos. Isto permite o compartilhamento da linha do cache entre interrupções especificadas e threads de aplicativo.

O valor da afinidade de interrupção para um número do IRQ específico é armazenado no arquivo associado **/proc/irq/*IRQ_NUMBER*/smp_affinity**, que pode ser visualizado e modificado pelo usuário root. O valor armazenado neste arquivo é um bit-mask hexadecimal representando todos os núcleos da CPU no sistema.

Por exemplo, para configurar a afinidade de interrupção para o driver Ethernet em um servidor com quatro núcleos CPU, determine primeiro o número IRQ associado ao driver Ethernet:

```
# grep eth0 /proc/interrupts
32:   0     140      45       850264      PCI-MSI-edge      eth0
```

Use o número IRQ para localizar o arquivo ***smp_affinity*** apropriado:

```
# cat /proc/irq/32/smp_affinity
f
```

O valor padrão para o smp_affinity é **f**, ou seja, o IRQ pode ser atentido em qualquer CPU no sistema. Configurar este valor para **1**, como se segue, significa que somente a CPU 0 pode atender esta interrupção:

```
# echo 1 >/proc/irq/32/smp_affinity
# cat /proc/irq/32/smp_affinity
1
```

Vírgulas podem ser utilizadas para delimitar valores ***smp_affinity*** para grupos discretos de 32 bits. Isto é necessário em sistemas com mais do que 32 núcleos. Por exemplo, o seguinte exemplo demonstra que o IRQ 40 é atendido em todos os núcleos de um sistema de núcleo de 64:

```
# cat /proc/irq/40/smp_affinity
ffffffff,ffffffff
```

Para atender ao IRQ 40 somente acima de 32 núcleos de um sistema de núcleo 64, você deve fazer o seguinte:

```
# echo 0xffffffff,00000000 > /proc/irq/40/smp_affinity
# cat /proc/irq/40/smp_affinity
ffffffff,00000000
```

**Nota**

Nos sistemas que suportam *direcionamento de interrupção*, modificar o ***smp_affinity*** de um IRQ define o hardware para que a decisão de atenter uma interrução com uma CPU específica seja feita em nível de hardware, sem intervenção do kernel.

## 4.4. Melhorias do NUMA no Red Hat Enterprise Linux 6

O Red Hat Enterprise Linux 6 inclui uma série de melhorias para aproveitar todo o potencial de hardware altamente escalável de hoje. Esta seção dá uma visão geral de alto nível das mais importantes melhorias de desempenho relacionados ao NUMA, fornecidos pela Red Hat Enterprise Linux 6.

### 4.4.1. Bare-metal e Otimizações de Escalabilidade

#### 4.4.1.1. Melhorias no aviso sobre a topologia

As seguintes melhorias permitem que o Red Hat Enterprise Linux detecte hardware de baixo nível e detalhes de arquiteturas, aprimorando sua habilidade para otimizar automaticamente o processamento em seu sistema.

**detecção de topologia aprimorada**

Isto permite que o sistema operacional detecte os detalhes do hardware de baixo nível (tais como CPUs lógicas, threads hiper, núcleos, sockets, nós de NUMA e tempos de acesso entre nós) durante a inicialização, e otimizar o processamento em seu sistema.

**agendador totalmente justo**

Este novo modo de agendamento assegura que o tempo de execução seja compartilhado entre

os processos elegíveis. Combinado a isto, a detecção da topologia permite os processos serem agendados nas CPUs dentro do mesmo soquete para evitar a necessidade por acesso de memória remota cara, e assegurar que o conteúdo do cache está preservado onde for possível.

**`malloc`**

`malloc` agora é otimizado para assegurar que as regiões da memória que estão alocadas a um processo estejam o mais próximas fisicamente o possível do núcleo no qual o processo está sendo executado. Isto aumenta a velocidade do acesso de memória.

**Alocação de buffer de E/S do skbuff**

Da mesma forma que o `malloc`, isto agora otimiza o uso de memória que esteja fisicamente próxima da CPU manuseando as operações de E/S como interrupções de dispositivo.

**afinidade de interruptção de dispositivo**

Informações gravadas pelos drivers de dispositivo sobre o qual a CPU manuseia quais interrupções podem ser usadas para restringir manuseio de interrupção em CPUs dentro do mesmo soquete físico, preservando afinidade de cache e limitando comunicação de soquete cruzado de alto volume.

#### 4.4.1.2. Melhorias em Sincronização de Multi-processador

Coordenar as tarefas entre processadores múltiplos requer operações de consomem tempo frequentes para garantir que os processos que estão executando em paralelo não comprometam a integridade de dados. O Red Hat Enterprise Linux inclui as seguintes melhorias para aprimorar desempenho nesta área:

**Bloqueios de Read-Copy-Update (RCU)**

Geralmente, 90% dos bloqueios são adquiridos somente para leitura. O bloqueio de RCU remove a necessidade de obter bloqueio de acesso exclusivo quando os dados que estiverem sendo acessados não sejam sendo modificados. Este modo de bloqueio é agora usado em alocação de memória de cache de página: bloqueio é agora usado somente para operações de alocação e desalocação.

**algorítimos per-CPU e per-socket**

Muitos algoritmos foram atualizados para realizar a coordenação de bloqueio entre CPUs cooperando na mesma soquete para permitir mais bloqueio refinado. Diversos spinlocks globais foram substituídos por métodos de bloqueio per-socket e zonas de alocador de memória atualizadas e listas de páginas de memórias relacionadas permietem alocação lógica para atravessar um subconjunto mais eficiente das estruturas de dados de mapeamento de memória ao executar operações de alocação ou desalocação.

### 4.4.2. Otimizaçãoes de Virtualização

Como o KVM utiliza a funcionalidade do kernel, os convidados virtualizados baseados em KVM se beneficiam imediatamente de todas as otimizações bare-metal. O Red Hat Enterprise Linux também inclui uma série de melhorias para permitir que os convidados virtualizados se aproximem do nível de desempenho de um sistema bare-metal. Essas melhorias se concentram no caminho de E/S no

armazenamento e acesso à rede, permitindo que as cargas de trabalho, mesmo intensivas, tais como banco de dados e arquivo de serviço, façam uso da implantação virtualizada. Melhorias específicas ao NUMA que melhoram o desempenho dos sistemas virtualizados incluem:

**Fixação de CPU**

Convidados virtuais podem ser vinculados para executar em um soquete específico para otimizar o uso do cache local e remover a necessidade de comunicações inter-soquete caras e acesso de memória remota.

**transparent hugepages (THP)**

Com o THP habilitado, o sistema realiza automaticamente as requisições de alocação de memória do NUMA para quantias contínuas grandes de memória, reduzindo a contenção de bloqueio e o número de operações de gerenciamento de memória do translation lookaside buffer (TLB) necessárias e gerando um crescente desempenho de até 20% em convidados virtuais.

**Implementação de E/S baseado no kernel**

O subsistema de E/S de convidado virtual foi implementado no kernel, reduzindo imensamente o custo de comunicação entre-nó e acesso de memória, evitando uma alta quantia de mudança de contexto e sobrecarga de sincronização e comunicação.

# Capítulo 5. Memória

Leia este capítulo para obter uma visão geral dos recursos de gerenciamento de memória disponíveis no Red Hat Enterprise Linux, e como utilizar estes recursos de gerenciamento para otimizar o uso de memória em seu sistema.

## 5.1. Buffer de Conversão Enorme à parte (Huge TLB)

Endereços de memória física são traduzidos para endereços de memória virtual como parte do gerenciamento de memória. A relação mapeada de endereços virtuais para físicas é armazenada numa estrutura de dados conhecida como a tabela de página. Como a leitura da tabela de páginas para cada endereço de mapeamento seria demorado e recursos caros, há um cache para os endereços usados recentemente. Esse cache é chamada de Buffer de Conversão à parte (TLB).

No entanto, a TLB só pode armazenar em cache tantos mapeamentos de endereço. Se um mapeamento de endereço solicitado não está na TLB, a tabela de páginas ainda deve ser lida para determinar o físico para o mapeamento de endereço virtual. Isto é conhecido como um "TLB miss". Aplicativos com grandes requerimentos de memória são mais propensos a ser afetados por falhas de TLB do que aplicativos com requerimento mínimo de memória, por causa da relação entre os seus requerimentos de memória e o tamanho das páginas usadas para mapeamentos de endereço de cache na TLB. Uma vez que cada falta envolve leitura da tabela de página, é importante para evitar a falha destes sempre que possível.

O Buffer Enorme de Conversão à parte (Huge TLB) permite que a memória seja gerenciada em grandes segmentos para que mais mapeamentos de endereços possam ficar em cache de uma só vez. Isto reduz a probabilidade das falhas de TLB, o que por sua vez aprimora o desempenho em aplicativos com grandes requerimentos de memória.

Informações sobre a configuração do HugeTLB pode ser encontrado na documentação do kernel: **`/usr/share/doc/kernel-doc-`*`version`*`/Documentation/vm/hugetlbpage.txt`**

## 5.2. Huge Pages e Transparent Huge Pages

A memória é gerenciada em blocos conhecidos como *pages*. Uma página tem 4096 bites. 1MB de memória é igual a 256 páginas; 1 GB de memória é igual a 256,000 páginas, etc. As CPUs possuem uma *unidade de gerenciamento de memória* que contém uma lista destas páginas, com cada página referenciada através de uma *entrada de tabela de página*.

Existem duas formas de habilitar o sistema para gerenciar grandes quantias de memória:

- Aumente o número de entradas de tabela de páginana unidade de gerenciamento de memória de hardware
- Aumente o tamanho da página

O primeiro método é caro, já que a unidade de gerenciamento de memória hardware de um processador moderno suporta apenas centenas ou milhares de entradas de tabela de página. Além disso, o hardware e os algoritmos de gerenciamento de memória que funcionam bem com milhares de páginas (megabytes de memória) podem ter dificuldade para realizar bem com milhões (ou até bilhões) de páginas. Isso resulta em problemas de desempenho: quando um aplicativo precisa usar mais páginas de memória do que a unidade de gestão de memória suporta, o sistema retorna mais lento, gerenciamento de memória baseado em software, o que faz com que todo o sistema funcionar mais devagar.

Red Hat Enterprise Linux 6 implementa o segundo método via uso de *huge pages*.

Colocando de uma forma simples, huge pages são blocos de memória de 2MB e 1GB de tamanho. As tabelas de páginas usadas pelas páginas de 2MB são adequadas para gerenciar gigabytes múltiplos de memória, onde as tabelas de páginas com 1GB são melhores para escalar para terabytes de memória.

Huge pages devem ser atribuídas durante a inicialização. Elas também são difíceis de gerenciar manualmente, e geralmente requerem mudanças significativas no códigos para serem utilizados de forma efetiva. Como tal, o Red Hat Enterprise 6 também implementou o uso do *transparent huge pages* (THP). O THP é uma camada de abstração que automatiza a maioria dos aspectos de criação, gerenciamento e uso de huge pages.

THP esconde muito da complexidade do uso do huge pages de administradores e desenvolvedores do sistema. Como o objetivo de THP é melhorar o desempenho, seus desenvolvedores (tanto da comunidade e Red Hat) já testaram e otimizaram THP através de uma ampla gama de sistemas, configurações, aplicativos e cargas de trabalho. Isto permite que as configurações padrão de THP melhorem o desempenho da maioria das configurações do sistema.

Note que o THP pode mapear atualmente somente regiões de memória anônimas, assim como espaços de heap e pilha.

## 5.3. Utilizando o Valgrind para o Uso de Memória de Perfil

**Valgrind** é um quadro que fornece instrumentação para binários do espaço do usuário. Ele vem com uma série de ferramentas que podem ser usadas para o perfil e analisar o desempenho do programa. As ferramentas apresentadas nesta seção fornecem uma análise que pode auxiliar na detecção de erros de memória, tais como o uso de memória não inicializada e alocação ou desalocação de memória impróprios. Todos estão incluídos no *valgrind* pacote, e pode ser executado com o seguinte comando:

```
valgrind --tool=toolname program
```

Substitua ***toolname*** pelo nome da ferramenta que você deseja usar (para perfil de memória, **memcheck**, **massif**, ou **cachegrind**), e ***program*** com o programa que você deseja realizar o perfil com o Valgrind. Tenha em mente que a instrumentação do Valgrind fará com que o seu programa execute mais vagarosamente do que o normal.

Uma visão geral das capacidades do Valgrind é fornecida emSeção 3.5.3, “Valgrind”. Maiores detalhes, incluindo informações sobre os plugins disponíveis para o Eclipse, estão incluídos no *Developer Guide*, disponível a partir do http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/. A documentação que acompanha pode ser visualizada com o comando **man valgrind** quando o pacote *valgrind* estiver instalado, ou encontrado nos seguintes locais:

- **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**, and
- **/usr/share/doc/valgrind-*version*/html/index.html**.

### 5.3.1. Uso de Memória de Perfil com o Memcheck

Memcheck é a ferramenta padrão do Valgrind, e pode ser executado com **valgrind*programa***, sem especificar **-- tool=memcheck** . Ele detecta e relata uma série de erros de memória que podem ser difíceis de detectar e diagnosticar, assim como o acesso à memória que não deve ocorrer, o uso de valores indefinidos ou não inicializado, memória heap liberada incorretamente, sobrepondo ponteiros, e vazamentos de memória. Os programas funcionam de dez à trinta vezes mais lentamente com Memcheck que quando executado normalmente.

Memcheck retorna erros específicos, dependendo do tipo de problemas que detectar. Estes erros são

descritos em detalhes nos documentos incluídos no Valgrind **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

Note-se que Memcheck só pode denunciar esses erros — não pode impedir que eles ocorram. Se o seu programa acessa a memória de uma forma que normalmente resultaria em uma falha de segmentação, a falha de segmentação ainda ocorre. No entanto, Memcheck irá registrar uma mensagem de erro imediatamente antes da falha.

Memcheck proporciona opções de linha de comando que podem ser usadas para focar o processo de verificação. Algumas das opções disponíveis são:

**--leak-check**
: Quando ativado, Memcheck procura por vazamentos de memória quando termina o programa cliente. O valor padrão é **summary**, que gera o número de vazamentos encontrados. Outros valores possíveis são **yes** e **full**, sendo que ambos dão detalhes de cada vazamento individual, e **no**, que desativa a verificação de vazamento de memória.

**--undef-value-errors**
: Quando ativado (definido para **yes**), Memcheck relata erros quando os valores indefinidos são usados. Quando desativado (definido para **no**), os erros de valor indefinido não são relatados. Isso é ativado por padrão. Desativar ele irá acelerar Memcheck.

**--ignore-ranges**
: Permite que o usuário especifique uma ou mais faixas que o Memcheck deva ignorar durante a verificação de endereçamento. Múltiplas faixas são delimitadas por vírgulas, por exemplo, **--ignore-ranges=0xPP-0xQQ,0xRR-0xSS**.

Para uma lista completa de opções consulte a documentação incluída em **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

### 5.3.2. Uso de Cache de Perfil com o Cachegrind

Cachegrind simula a interação do seu programa com a hierarquia de cache de uma máquina e (opcionalmente) ramificação preditora. Ele controla o uso da instrução de primeiro nível simulada e caches de dados para detectar a interação do código pobre com este nível de cache; e o cache de último nível, mesmo que seja um segundo ou terceiro nível de cache, a fim de controlar o acesso à principal memória. Como tal, os programas são executados com Cachegrind funcionam 20-100 vezes mais lentos do que quando executados normalmente.

Para executar o Cachegrind, execute o seguinte comando, substituindo o ***program*** pelo programa que você deseja realizar o perfil com o Cachegrind.

```
# valgrind --tool=cachegrind program
```

O Cachegrind pode reunir as seguintes estatísticas para todo o programa, e para cada função no programa:

- as leituras de Cache de instrução de primeiro nível (ou as instruções executadas) e falta de leituras, e falta de leitura de instrução de cache de último nível;
- leituras de cache de dados (ou leituras de memória), falta de leituras, e falta de leituras de dados do cache de último nível;

- gravações de cache de dados (ou gravações de memória), falta de gravações, e falta de gravações de último nível
- ramificações condicionais executadas e mal previstas; e
- ramificações indiretas executadas e mal previstas.

O Cachegrind imprime as informações de resumo sobre estas estatísticas no console e grava informações mais detalhadas de perfil em um arquivo (**cachegrind.out.*pid***por padrão, onde ***pid*** é o ID do processo do programa no qual você executou o Cachegrind). Este arquivo pode ser processado mais tarde acompanhado da ferramenta **cg_annotate**, como abaixo:

```
# cg_annotate cachegrind.out.pid
```

**Nota**

**cg_annotate** pode produzir linhas maiores do que 120 caracteres, dependendo do comprimento do caminho. Para tornar a saída mais clara e fácil de ler, recomendamos fazer a sua janela de terminal pelo menos desta largura antes de executar o comando acima mencionado.

Você também pode comparar os arquivos de perfil criados pelo Cachegrind para facilitar o desempenho do programa da tabela antes e depois de uma mudança. Para fazer isto, use o comando **cg_diff** substituindo ***first*** pelo arquivo de resultado do perfil inicial, e ***second*** pelo arquivo de resultado de perfil subsequente:

```
# cg_diff first second
```

Este comando produz um arquivo de resultado combinado, que pode ser visualizado em mais detalhes com o **cg_annotate**.

Cachegrind suporta um número de opções para focar seu resultado. Algumas das opções disponíveis são:

**--I1**
: Especifica o tamanho, associatividade e tamanho da linha do cache de instrução de primeiro nível, separado por vírgulas: **--I1=*size*,*associativity*,*line size***.

**--D1**
: Especifica o tamanho, associatividade e tamanho da linha do cache de dados de primeiro nível, separado por vírgulas: **--D1=*size*,*associativity*,*line size***.

**--LL**
: Especifica o tamanho, associatividade e tamanho da linha do cache último nível, separado por vírgulas: **--LL=*size*,*associativity*,*line size***.

**--cache-sim**
: Habilita ou desabilita a coleção de acesso a cache e contas faltando. O valor padrão é **yes** (habilitado).

  Note que desabilitar este e **--branch-sim** deixa o Cachegrind sem informações para coletar.

**--branch-sim**

Habilita ou desabilita a coleção de instruções de ramificação e contas mal previstas. Isto é definido para **no** (desabilitado) por padrão, pois ele desacelera o Cachegrind em aproximadamente 25 porcento.

Note que desabilitar este e **--cache-sim** deixa o Cachegrind sem informações para coletar.

Para uma lista completa de opções consulte a documentação incluída em **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

### 5.3.3. Heap do Perfil e Espaço de Pilha com Massif

Massif mede o espaço de pilha usado por um programa específico, tanto o espaço útil quanto qualquer espaço adicional alocado para fins de contabilidade e alinhamento. Ele pode ajudá-lo a reduzir a quantidade de memória utilizada pelo seu programa, o que pode aumentar a velocidade do seu programa, e reduzir a probabilidade de que seu programa irá esgotar o espaço de troca da máquina em que ele executa. Massif também pode fornecer detalhes sobre quais partes do seu programa são responsáveis pela alocação de memória heap. Programas executados com Massif são executados com cerca de vinte vezes mais lentidão do que a sua velocidade de execução normal.

Para realizar o perfil do uso do heap de um programa, especifique o **massif** como uma ferramenta Valgrind que você deseja utilizar:

```
# valgrind --tool=massif program
```

Dados de perfil coletados pelo Massif são gravados em um arquivo, que por padrão é chamado de **massif.out.*pid***, onde ***pid*** é o ID de processo do ***programa*** especificado.

Estes dados de perfil também podem ser grafados com o comando **ms_print**, assim como:

```
# ms_print massif.out.pid
```

Isso produz um gráfico que mostra o consumo de memória sobre a execução do programa, e informações detalhadas sobre os locais responsáveis pela alocação em vários pontos do programa, incluindo no ponto de alocação de memória de pico.

Massif fornece um número de opções de linha de comando que podem ser usados para dirigir a saída da ferramenta. Algumas das opções disponíveis são:

**--heap**

Especifica se deve realizar perfil de heap. Este valor padrão é **yes**. O perfil do Heap pode ser desabilitado configurando esta opção para **no**.

**--heap-admin**

Especifica o número de bytes por bloco a usar para a administração quando um perfil de heap for habilitado. O valor padrão é **8** bytes por bloco.

**--stacks**

Especifica se deve criar o perfil da pilha. O valor padrão é **no** (desabilitado). Para habilitar

perfis de pilha, definir esta opção para **yes**, mas esteja ciente de que isso desacelera o Massif. Observe também que Massif assume que a pilha principal tem tamanho zero na inicialização para indicar melhor o tamanho da porção da pilha sobre a qual o programa que está sendo perfilado tem controle.

**--time-unit**

Especifica a unidade de tempo utilizado para a criação de perfil. Há três valores válidos para esta opção: instruções executadas (**i**), o valor padrão, que é útil na maioria dos casos, tempo real (**ms**, em milissegundos), que pode ser útil em certos casos, e bytes alocados / desalocado na pilha e / ou stack (**B**), que é útil para os programas de muito curto prazo, e para fins de teste, porque é o mais reproduzível em máquinas diferentes. Esta opção é útil ao criar gráficos de resultado de Massif com **ms_print**.

Para uma lista completa de opções consulte a documentação incluída em **/usr/share/doc/valgrind-*version*/valgrind_manual.pdf**.

## 5.4. Ajuste de Capacidade

Leia esta seção para obter um esboço de memória, kernel e capacidade do sistema de arquivos, os parâmetros relacionados a cada um, e os dilemas para ajustar esses parâmetros.

Para definir esses valores temporariamente durante o ajuste, copie o valor desejado para o arquivo adequado no sistema de arquivos proc. Por exemplo, para definir ***overcommit_memory*** temporariamente para **1**, execute:

```
# echo 1 > /proc/sys/vm/overcommit_memory
```

Note que o caminho para o parâmetro no sistema de arquivos proc varia dependendo do sistema afetado pela mudança.

Para definir estes valores de forma persistente, você precisará utilizar o comando **sysctl**. Para maiores detalhes, consule o *Deployment Guide*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

**Ajustes de Memória relacionados com a Capacidade**

Cada um dos parâmetros a seguir está localizado sob **/proc/sys/vm/** no sistema de arquivo do proc.

***overcommit_memory***

Define as condições que determinam se uma requisição de memória grande é aceita ou negada. Existem três valores possíveis para este parâmetro:

- **0** — A configuração padrão. O kernel realiza o overcommit heurístico do kernel, manuseando isto através da estimativa da quantia de memória disponível e queda de requisições que sejam totalmente inválidas. Infelizmente, como a memória é alocada utilizando um heurístico ao invés de um algorítimo preciso, esta configuração pode as vezes permitir que a memória disponível no sistema seja sobrecarregada.
- **1** — O kernel não faz manuseio de overcommit de memória. Sob esta configuração, o potencial para sobrecarga de memória aumenta, como também o desempenho para tarefas intensivas de memória.
- **2** — The kernel nega requisições para memórias iguais ou maiores do que a quantia de

swap total disponível e a porcentagem de RAM física especificada em ***overcommit_ratio***. Esta configuração é a melhor se você quiser diminuir o risco de overcommit de memória.

**Nota**

Esta configuração é recomendada somente para sistemas com áreas de swap maiores do que sua memória física.

***overcommit_ratio***

Especifica a porcentagem da RAM física considerada quando o ***overcommit_memory*** está definido para **2**. O valor padrão é **50**.

***max_map_count***

Define o número máximo de áreas de mapa de memória que um processo pode utilizar. Na maioria dos casos, o valor padrão de **65530** é adequado. Aumente este valor se seu aplicativo precisar mapear mais do que este número de arquivos.

***nr_hugepages***

Define o número de hugepages configurado no kernel. O valor padrão é 0. É possível alocar (ou desalocar) hugepages, somente se existirem páginas livres contínuas fisicamente no sistema. As páginas reservadas por este parâmetro não podem ser utilizadas para outros propósitos. Mais informações podem ser obtidas da documentação instalada: **/usr/share/doc/kernel-doc-*kernel_version*/Documentation/vm/hugetlbpage.txt**

### Ajustes de Kernel relacionados com a Capacidade

Cada um dos parâmetros a seguir está localizado sob o sistema de arquivo do proc **/proc/sys/kernel/** no sistema de arquivo do proc.

***msgmax***

Define o tamanho permitido em bites de qualquer mensagem única em uma fila de mensagem. Este valor não deve exceder o tamanho da fila (***msgmnb***). O valor padrão é **65536**.

***msgmnb***

Define o tamanho máximo em bites de uma única fila de mensagem. O valor padrão é **65536** bytes.

***msgmni***

Define o número máximo de identificadores de filas de mensagens (e portanto o número máximo de filas). O valor padrão em máquinas com arquitetura 64-bit é de **1985**; for 32-bit architecture, the default value is **1736**.

***shmall***

Define a quantia total de memória compartilhada em bites que possa ser utilizada no sistema

de uma só vez. O valor padrão para máquinas com arquitetura de 64-bit é de **4294967296**; para arquiteturas 32-bit o padrão é de **268435456**.

***shmmax***

Define o segmento máximo de memória compartilhada pelo kernel, em bites. O valor padrão em máquinas com arquitetura de 64-bit é de **68719476736**; para arquitetura de 32-bit, o valor padrão é de **4294967295**. Note, no entanto, que o kernel suporta valores muito maiores do que este.

***shmmni***

Define o número máximo de amplitude de sistema de sementos de memória compartilhada. O valor padrão é de **4096**em ambas arquiteturas de 64-bit e 32-bit.

***threads-max***

Define o número máximo de amplitude de sistema de discussões (tarefas) a serem utilizadas pelo kernel de uma só vez. O valor padrão é igual ao valor ***max_threads*** do kernel. A formula em uso é:

```
max_threads = mempages / (8 * THREAD_SIZE / PAGE_SIZE )
```

O valor mínimo de***threads-max*** é de **20**.

### Ajustes de Sistema de Arquivo Relacionada com a Capacidade

Cada um dos parâmetros a seguir está localizado sob o sistema de arquivo proc **/proc/sys/fs/**.

***aio-max-nr***

Define o número máximo permitido de eventos em todos os contextos de E/S assíncrona. O valor padrão é de **65536**.Note que ao modificar este valor, você não pré-aloca ou redimensiona qualquer estrutura de dados do kernel.

***file-max***

Lista o número máximo de manuseio de arquivo que o kernel aloca. O valor padrão coincide com o valor de ***files_stat.max_files*** no kernel, o qual está definido para o valor maior entre os **(mempages * (PAGE_SIZE / 1024)) / 10**, ou **NR_FILE** (8192 in Red Hat Enterprise Linux). Aumentar este valor pode resolver erros causados pela falta de manuseios de arquivos disponíveis.

### Ajustes Out-of-Memory Kill

Out of Memory (OOM) se refere a um estado de computação onde todas as memórias, incluindo o espaço swap, foi alocada. Por padrão, esta situação faz com que o sistema trave e pare de funcionar como deveria. No entanto, configurar o parâmetro ***/proc/sys/vm/panic_on_oom*** para **0** instrui o kernel a chamar a função **oom_killer** quando ocorrer o OOM. Geralmente o **oom_killer** pode eliminar processos invasores e o sistema sobrevive.

O parâmetro a seguir pode ser definido por processo, fornecendo maior controle sobre quais

processos são eliminados pela função **oom_killer**. Está localizada em ***/proc/pid/*** no sistema de arquivo do proc, onde ***pid*** é o ID do processo.

***oom_adj***

Define um valor a partir de **-16** até **15** que ajuda a determinar o ***oom_score*** de um processo. Quanto maior o valor do ***oom_score*** maior a probabilidade do processo ser eliminado pelo **oom_killer**. Configurar um valor ***oom_adj*** de **-17** desabilita o **oom_killer** para aquele processo.

**Importante**

Qualquer processo gerado pelo processo ajustado, irá herdar o ***oom_score*** daquele processo. Por exemplo, se um processo **sshd** é protegido da função **oom_killer** , todos os processos iniciados pela sessão SSH também serão protegidas. Isto pode afetar a habilidade das funções do **oom_killer** para salvar o sistema se ocorrer um OOM.

## 5.5. Ajustando Memória Virtual

A memória virtual é geralmente consumida pelos processos, caches de sistema de arquivo e kernel. O uso da memória virtual depende de uma série de fatores, que podem ser afetados pelos seguintes parâmetros:

***swappiness***

Um valor de 0 à 100 que controla o grau para o qual o sistema altera. Um valor alto dá prioridade ao desempenho do sistema, alterando os processos de forma agressiva fora da memória física quando eles não estão ativos. Um valor baixo dá prioridade à interação e evita processos de alteração fora da memória física o quanto de tempo for possível, o que diminui a latência de resposta. O valor padrão é **60**.

***min_free_kbytes***

O número mínimo de kilobytes para manter livre em todo o sistema. Este valor é usado para calcular um valor de marca d'água para cada zona de baixa memória, que recebem um número de páginas livres reservadas proporcionalmente ao seu tamanho.

**Valores extremos podem quebrar seu sistema**

Seja cauteloso ao configurar este parâmetro, pois tanto valores muito baixos como muito altos podem causar danos.
Configuração ***min_free_kbytes*** muito baixo previne o sistema de reclamar memória. Isto pode resultar em travamento de sistema e processos múltiplos de OOM-killing.
No entanto, configurar este parâmetro para um valor que seja muito alto (5-10% do total de memória de sistema) causará uma falta de memória em seu sistema imediatamente. O Linux foi criado para utilizar todas as RAM disponíveis para realizar um cache dos dados de sistema de arquivo. Configurar um valor alto de ***min_free_kbytes*** resulta em uma perda de tempo quando o sistema reclama memória.

***dirty_ratio***

Define um valor de porcentagem. Limpeza de dados sujos inicia com (via **pdflush**) quando os dados sujos comprimem esta porcentagem da memória de sistema total. O valor padrão é **20**.

***dirty_background_ratio***

Define um valor de porcentagem. Limpeza de dados sujos inicia no pano de fundo (via **pdflush**) quando os dados sujos comprimem esta porcentagem da memória de sistema total. O valor padrão é **10**.

***drop_caches***

Configurar este valor para **1**, **2**, or **3** faz com que o kernel derrube diversas páginas de combinação cache e cache de slab.

**1**

O sistema invalida e libera todas as memórias de cache de página.

**2**

O sistema libera toda a memória não utilizada de cache de slab.

**3**

O sistema libera toda a memória de cache de página e cache de slab.

Esta não é uma operação destrutiva. Como os objetos sujos não podem ser liberados, recomenda-se executar o **sync** antes de configurar este valor de parâmetro.

**Importante**

O uso do ***drop_caches*** para liberar memória não é recomendado em um ambiente de produção.

Para definir estes valores temporariamente durante o ajuste, copie o valor desejado no arquivo apropriado no sistema de arquivo proc. Por exemplo, para definir ***swappiness*** temporariamente para **50**, execute:

```
# echo 50 > /proc/sys/vm/swappiness
```

Para configurar este valor persistentemente, você irá precisar usar o comando **sysctl** . Para mais informações consulte o *Deployment Guide*, disponível em http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

# Capítulo 6. Entrada/Saída

## 6.1. Recursos

O Red Hat Enterprise Linux 6 apresenta diversas melhorias de desempenho na pilha de E/S:

- Discos de estado sólido (SSDs) agora são reconhecidos automaticamente, e o desempenho do agendador de E/S é ajustado para tirar vantagem das E/S altas por segundo (IOPS) que estes dispositivos podem realizar.
- Foi adicionado um suporte descartado no kernel para reportar classes de blocos não utilizadas para armazenamento adjacente. Isto ajuda o SSDs com seus algorítimos de nivelamento por uso. Isto também ajuda o armazenamento que suporta provisionamento de blocos lógicos (um tipo de espaço de endereço virtual para armazenamento) po manter as marcações mais perto na quantia atual do armanzenamento em uso.
- A implementação da barreira do sistema de arquivo foi sobreposta no Red Hat Enterprise Linux 6.1 para torná-la mais útil.
- **`pdflush`** foi substituído pelas discussões de flusher por dispositivo de backup, que aprimora muito a escalabilidade do sistema em configurações com contas LUN grandes.

## 6.2. Análises

O ajuste do desempenho de pilha de armazenamento requer um conhecimento de como os dados fluem através do sistema, assim como conhecimento profundo dos armazenamentos adjacentes e como ele funciona sob cargas de trabalho que variam. Isto também requer um conhecimento do próprio ajuste da carga de trabalho

Sempre que você implantar um novo sistema, é uma boa idéia analisar o armazenamento de baixo para cima. Inicie com LUNs e discos brutos e avalie seus desempenhos utilizando E/S direta que cruza o cache da página do kernel). Isto é o teste mais básico que você pode realizar e será padrão pelo qual você medir o desempenho de E/S na pilha. Inicie com um gerador de carga de trabalho básica (tal como o **`aio-stress`**) que produz leituras e gravações aleatórias e sequenciais em toda a variedade de tamanhos da E/S e profundidade de fila.

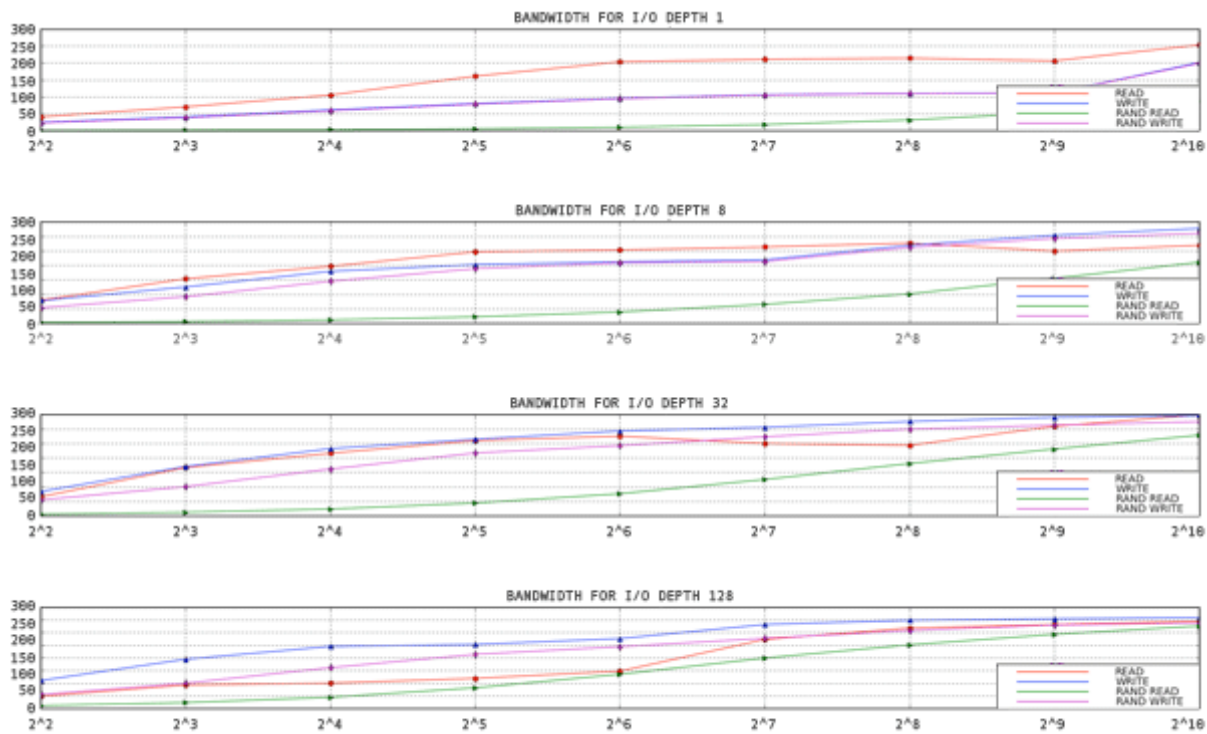
Segue abaixo um gráfico das séries de execuções **`aio-stress`**, cada qual desempenhando quatro estágios: gravações sequenciais, leituras sequenciais, gravações aleatórias e leituras aleatórias. Neste exemplo, a ferramenta é configurada para executar na classe de tamanho de histórico (o x axis) e profundidade de fila (uma por gráfico). A profundidade da fila representa o número total de operações de E/S em progresso em um determinado período.

O y axis mostra a largura da banda em megabytes por segundo. O x axis mostra o Tamanho da E/S em kilobytes.

**Figura 6.1. aio-stress output for 1 thread, 1 file**

Note como a linha produtividade tem a tendência do canto esquerdo inferior para o direito superior. Também note que, para um dado tamanho de histórico, você pode obter mais produtividade do armazenamento aumentando o número de E/S em progresso.

Ao executar essas cargas de trabalho simples em seu armazenamento, você compreenderá como o seu armazenamento realiza sob carga. Mantenha os dados gerados por esses testes de comparação quando analisar as cargas de trabalho mais complexos.

Se você utilizar mapeador de dispositivo ou md, adicione para a camada no próximo e repita os testes. Se houver uma grande perda de desempenho, assegure-se que ele é esperado, ou que pode ser explicado. Por exemplo, uma queda de desempenho pode ser esperada se uma camada de checksumming raid for adicionada à pilha. Trava inesperada no desempenho inesperado pode ser causada por operações de E/S desalinhadas. Por padrão, o Red Hat Enterprise Linux alinha partições e metadados do mapeador de dispositivo adquadamente. No entanto, nem todos os tipos de armazenamento reportam seus alinhamentos adequados, e por isso pode exigir um ajuste manual.

Após adicionar o mapeador de disposibito ou camada md, adicione o sistema de arquivo em cima do dispositivo de bloco e teste nele, ainda utilizando a E/S direta. Compare resultados com os testes anteriores e assegure-se de que você entende as discrepâncias. E/S de Gravações diretas geralmente funciona melhor em arquivos pré-alocados, para assegurar que você pré-alocou arquivos antes de testar o desempenho.

Geradores de carga de trabalho sintéticos que você pode achar úteis incluem:

- **aio-stress**
- **iozone**
- **fio**

## 6.3. Ferramentas

Existem diversas ferramentas disponíveis para ajduar nos problemas com o desempenho de diagnose no subsistema de E/S. O **vmstat** provê uma visão geral do desempenho do sistema. As colunas a seguir são as mais relevantes para a E/S: ***si*** (swap in), ***so*** (swap out), ***bi*** (block in), ***bo*** (block out), e ***wa*** (E/S tempo de espera). ***si*** e ***so*** são úteis quando o espaço swap estiver no mesmo dispositivo que sua partição de dados, e como um indicador de pressão de memória generalizada. O ***si*** e ***bi*** são operações de leitura, enquanto o ***so*** e ***bo*** são operações de gravação. Cada uma destas categorias é reportada em kilobytes. O ***wa*** é o tempo ocioso, ele indica qual porção de fila de execução está bloqueada esperando pela E/S ser concluída.

Analisar seu sistema com o **vmstat**lhe dárá uma idéia de se o subsistema da E/S deve ser responsável ou não pelos problemas de desempenho. As colunas ***free***, ***buff***, e ***cache*** também valem ser observadas. O valor ***cache*** crescendo junto do valor ***bo***seguido de uma caída de sistema ***cache*** e um aumento no ***free*** indica que o sistema está realizando um write-back e invalidação do cache da página.

Observe que os números de E/S reportados pelo **vmstat** são agregados de todos os E/S em todos os dispositivos. Depois que você determinou que pode existir o gap de desempenho no subsistema de E/S, você pode examinar o problema mais detalhadamente com o **iostat**, que dividirá a reportagem da E/S por dispositivo. Você também pode recuperar mais informações detalhadas, tal como a média do tamanho da requisição, o número e gravações por segundo e a quantia de mesclagem de E/S que está ocorrendo.

Utilizando a média de tamanho de requisição e a média de tamanho de fila **avgqu-sz**), você poderá estimar sobre como o armazenamento deveria funcionar utilizando gráficos que você gerou ao caracterizar o desempenho de seu armazenamento. Algumas generalizações se aplicam: por exemplo, se a media de tamanho de requisição é de 4KB e a média de tamanho de fila é de 1, a produtividade pode não ser tão útil.

Se os números de desempenho não mapeiam o desempenho que você espera, você pode realizar uma análise mais refinada, com o **blktrace**. O suite de utilitários do **blktrace** fornece informações refinadas sobre quanto tempo se gasta no subsistema de E/S. O resultado do **blktrace** é um conjunto de arquivos de traço binários que podem ser processados posteriormente por outros utilitários tal como **blkparce**.

**blkparse** é o utilitário companheiro do**blktrace**. Ele lê resultados brutos do traço e produz uma versão textual resumida.

Este é um exemplo do resultado do **blktrace** :

```
8,64   3        1     0.000000000  4162  Q  RM 73992 + 8 [fs_mark]
8,64   3        0     0.000012707     0  m   N cfq4162S / alloced
8,64   3        2     0.000013433  4162  G  RM 73992 + 8 [fs_mark]
8,64   3        3     0.000015813  4162  P   N [fs_mark]
8,64   3        4     0.000017347  4162  I   R 73992 + 8 [fs_mark]
8,64   3        0     0.000018632     0  m   N cfq4162S / insert_request
8,64   3        0     0.000019655     0  m   N cfq4162S / add_to_rr
8,64   3        0     0.000021945     0  m   N cfq4162S / idle=0
8,64   3        5     0.000023460  4162  U   N [fs_mark] 1
8,64   3        0     0.000025761     0  m   N cfq workload slice:300
8,64   3        0     0.000027137     0  m   N cfq4162S / set_active wl_prio:0
wl_type:2
8,64   3        0     0.000028588     0  m   N cfq4162S / fifo=(null)
8,64   3        0     0.000029468     0  m   N cfq4162S / dispatch_insert
8,64   3        0     0.000031359     0  m   N cfq4162S / dispatched a request
8,64   3        0     0.000032306     0  m   N cfq4162S / activate rq, drv=1
8,64   3        6     0.000032735  4162  D   R 73992 + 8 [fs_mark]
8,64   1        1     0.004276637     0  C   R 73992 + 8 [0]
```

Como você pode ver, o resultado é denso e difícil de se ler. Você pode dizer quais processos são responsável a fim de expressar E/S para seu dispositivo, o qual é utilizado mas o **blkparse** pode lhe fornecer informações adicionais de formato fácil em seu sumário. As informações do sumário do **blkparse** são impressas no final do resultado:

```
Total (sde):
Reads Queued:           19,       76KiB  Writes Queued:     142,183,  568,732KiB
Read Dispatches:        19,       76KiB  Write Dispatches:   25,440,  568,732KiB
Reads Requeued:          0               Writes Requeued:       125
Reads Completed:        19,       76KiB  Writes Completed:   25,315,  568,732KiB
Read Merges:             0,        0KiB  Write Merges:      116,868,  467,472KiB
IO unplugs:         20,087               Timer unplugs:           0
```

O resumo demonstra a média de taxas de E/S, mescla de atividades e compara a carga de trabalho de leitura com a carga de trabalho de gravação. Para a maior parte, no entanto, o resultado do **blkparse** é muito volumoso para ser útil sozinho. Felizmente, existem diversas ferramentas para assistir na visualização de dados.

**btt** fornece uma análise da quantia de tempo que a E/S gasta em diferentes áreas da pilha de E/S. São estas as áreas:

- Q — Uma E/S de bloco é Enfileirada
- G — Obtenha Requisição

  Uma E/S de bloco enfileirada recentemente, não foi um candidato para mesclar com qualquer requisição existente, portanto uma requisição de camada de bloco nova é alocada.
- M — Uma E/S de bloco é Mesclada com uma requisição existente.
- I — Uma requisição é inserida na fila dos dispositivos.
- D — Uma requisição é enviada ao Dipositivo
- C — Uma requisição é concluída pelo motorista
- P — A fila do dispositivo de bloco é Ligada, para permitir a agregação de requisições.
- U — A fila de dispositivo é Desligada, permitindo que as requisições agregadas sejam enviadas ao dispositivo.

**btt** divide em tempo gasto em cada uma destas áreas, assim como o tempo gasto transicionando entre eles, assim como:

- Q2Q — tempo entre requisições enviadas à camada de bloco.
- Q2G —quanto tempo leva do tempo que uma E/S de bloco é enfileirada até o tempo que ela obtém uma requisição alocada para isto.
- G2I — quanto tempo leva desde que um pedido é atribuído até o momento em que é inserido na fila do dispositivo
- Q2M — quanto tempo leva desde quando um bloco de E/S foi enfileiirado até que se mescle com um pedido existente
- I2D — quanto tempo demora a partir do momento que um pedido é inserido na fila do dispositivo até que seja realmente emitido para o dispositivo
- M2D — quanto tempo leva desde que um bloco E/S seja mesclado com um pedido de saída até que o pedido seja emitido para o dispositivo
- D2C —tempo de serviço da requisição por dispositivo
- Q2C — tempo total gasto em camada de bloco para uma requisição

Você pode deduzir muito de uma carga de trabalho a partir da tabela acima. Por exemplo, se Q2Q é muito maior do que Q2C, isso significa que o aplicativo não está emitindo E/S em sucessão rápida. Assim, todos os problemas de desempenho que você tem podem não estar relacionados ao subsistema de E/S. Se D2C é muito elevado, então o dispositivo está demorando muito para servir as requisições. Isto pode indicar que o dispositivo simplesmente está sobrecarregado (que pode ser devido ao fato de que é um recurso compartilhado), ou pode ser devido à carga de trabalho enviada para o dispositivo ser sub-óptima. Se Q2G é muito alto, isso significa que há um grande número de solicitações na fila ao mesmo tempo. Isso pode indicar que o armazenamento é incapaz de manter-se com a carga de E/S.

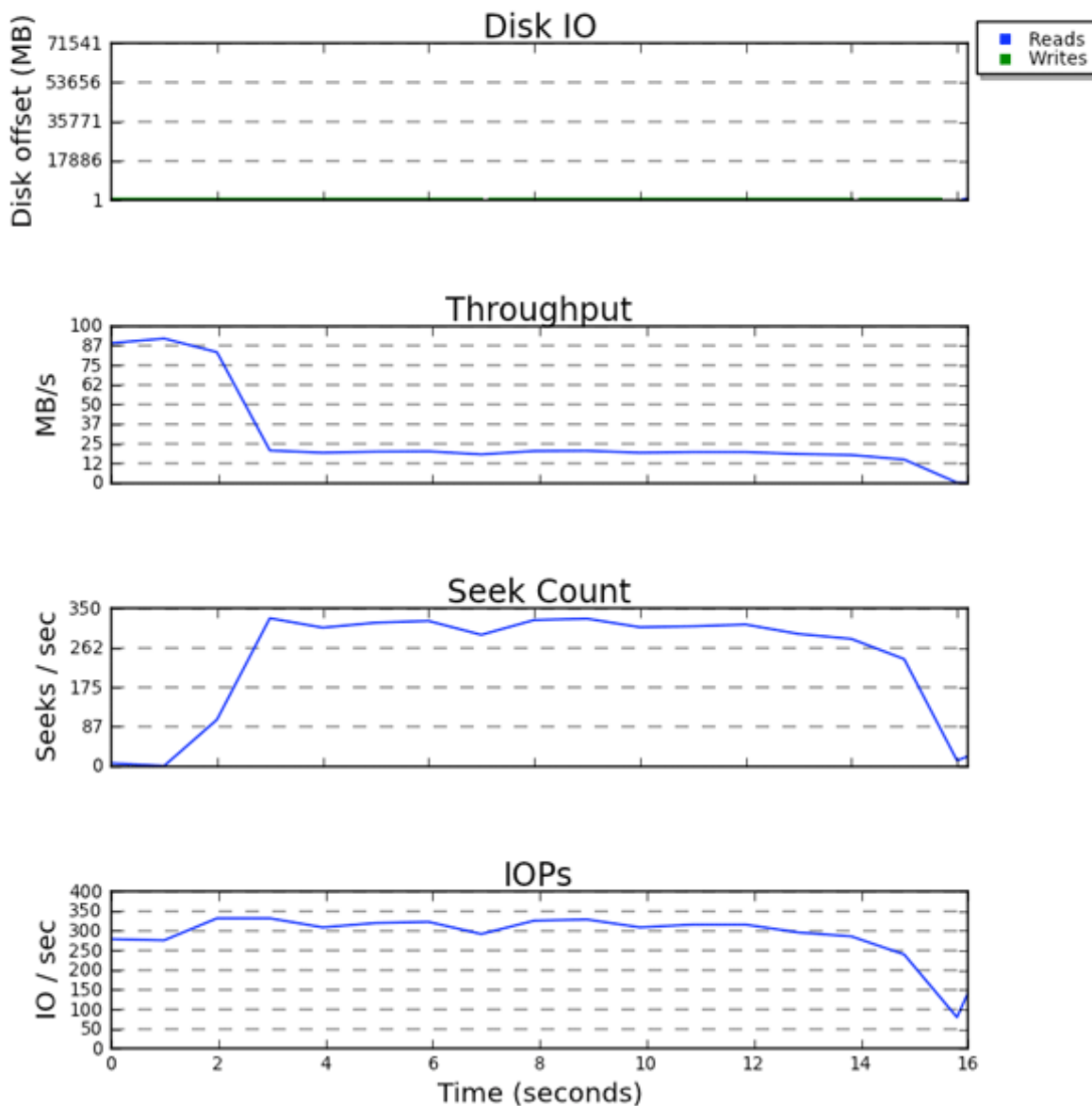
Finalmente, o **seekwatcher** consome dados de binários do **blktrace**e gera conjuntos de plotagens, incluindo o Logical Block Address (LBA), rendimento, procuras por segundo, e I/Os Per Second (IOPS).

| Avg Seeks/s | Avg MB/s | Avg IO/s | Run time (s) |
|---|---|---|---|
| 252.57 | 31.72 | 305.99 | 16.21 |

**Figura 6.2. Exemplo de resultado do seekwatcher**

Todos os lotes utilizam o tempo como o eixo X. A plotagem mostra as leituras e gravações em cores diferentes. É interessante observar a relação entre os gráficos de rendimento e procura/por seg. Para o armazenamento sensível à busca, existe uma relação inversa entre as duas plotagens. O gráfico IOPS é útil se, por exemplo, você não está recebendo o rendimento que se espera de um dispositivo, mas você está batendo suas limitações IOPS.

## 6.4. Configuração

Uma das primeiras decisões que você precisará tomar é escolher qual agendador de E/S utilizar. Esta seção provê uma visão geral de cada um dos agendadores principais para ajudá-lo a decidir qual o melhor para sua carga de trabalho.

### 6.4.1. Completely Fair Queuing (CFQ)

CFQ tenta fornecer alguma justiça nas decisões de agendamento de E/S com base no processo que iniciou a E/S. Três classes de agendamento diferentes são fornecidos: em tempo real (Reat Time - RT), de melhor esforço (Best-Effort -BE), e ocioso. A classe de agendamento pode ser atribuída manualmente a um processo com o comando **`ionice`**, ou programaticamente atribuída através da chamada de sistema **`ioprio_set`**. Por padrão, os processos são colocados na classe de agendamento de melhor esforço. As classes de agendamento em tempo real e de melhor esforço são subdivididos em oito prioridades de E/S dentro de cada classe, sendo a prioridade 0 a mais alta e 7 a mais baixa. Processos na classe de agendamento em tempo real estão programados muito mais agressivamente do que em processos em melhor esforço ou inativo, portanto, qualquer E/S de tempo real programada é sempre realizada antes da E/S de melhor esforço ou ocioso. Isto significa que a prioridade de E/S em tempo real pode desaparecer com as classes de melhor esforço e ociosas. O agendamento do melhor-esforço é a classe de agendamento padrão e 4 é a prioridade padrão nesta classe. Processos na classe de agendamento de repouso são apenas notados quando não há nenhuma outra E/S pendente no sistema. Assim, é muito importante definir apenas a classe de agendamento de E/S de um processo para ocioso se a E/S do processo não for necessária para fazer quaisquer progressos futuros.

CFQ fornece equidade pela atribuição de um intervalo de tempo para cada um dos processos realizando E/S. Durante seu intervalo de tempo, um processo pode ter (por padrão), até 8 pedidos em vôo de cada vez. O agendador tenta antecipar se um aplicativo emitirá mais E/S em um futuro próximo com base em dados históricos. Caso seja esperado que um processo irá emitir mais E/S, então o CFQ será ocioso, esperando por aquela E/S, mesmo se houver E/S de outros processos à espera de ser emitido.

Por causa da ociosidade realizada pelo CFQ, que muitas vezes não é uma boa opção para hardware que não sofrem de uma grande penalidade de busca, como matrizes de armazenamento externos rápidas ou discos de estado sólido. Se o uso do CFQ em tal armazenamento é uma exigência (por exemplo, se você também deseja usar o agendador cgroup de E/S de peso proporcional), você terá que ajustar algumas configurações para melhorar o desempenho do CFQ. Defina os seguintes parâmetros nos arquivos de mesmo nome localizada no **`/sys/block/`*`device`*`/queue/iosched/`**:

```
slice_idle = 0
quantum = 64
group_idle = 1
```

Quando o ***`group_idle`*** está configurado para 1, existe ainda o potencial para interrupções de E/S (onde o armazenamento do backend não é cheio devido à ociosidade). No entanto, estas interrupções serão menos frequentes do que a ociosidade em cada fila no sistema.

CFQ é um agendador de E/S de conservação de folga, o que significa que pode ser ocioso, mesmo quando existem solicitações pendentes (como vimos acima). O empilhamento de programadores de conservação de folga pode introduzir grandes latências no caminho de E/S. Um exemplo deste tipo de empilhamento é o uso do CFQ além de um controlador RAID hardware baseado em host. O controlador RAID pode implementar o seu próprio agendador de conservação de folga, causando atrasos em dois níveis na pilha. Conservação de folga funciona melhor quando têm dados suficientes para basear as suas decisões. No caso de empilhamento desses algoritmos de escalonamento, o agendador mais inferior só vai ver o que o agendador superior envia para baixo. Assim, a camada inferior verá um padrão de E/S que não é representativo do volume de trabalho real.

### Ajustáveis

***back_seek_max***

Buscas revertidas são geralmente ruins para o desempenho, pois podem incorrer em maiores atrasos no reposicionamento dos cabeçalhos do que as buscas normais. No entanto, o CFQ ainda vai realizá-las, se elas forem pequenas o suficiente. Este ajuste controla a distância máxima em KB que o agendador de E/S permitirá a procura revertida. O padrão é **16** KB.

***back_seek_penalty***

Devido à ineficiência da procura revertida, uma penalidade está associada a cada um. A pena é um multiplicador, por exemplo, considere a posição da cabeça do disco de 1024 KB. Suponha que existem dois pedidos na fila, um de 1008KB e outro em 1040KB. Os dois pedidos estão equidistantes da posição da cabeça atual. No entanto, depois de aplicar a pena de procura invertida (padrão: 2), a requisição em uma posição futura em discos estará agora duas vezes mais perto do que as requisições anteriores. Assim, a cabeça moverá para frente.

***fifo_expire_async***

Este ajuste controla quanto tempo uma requisição assíncrona (gravação de buffer) pode ficar sem serviços. Após o tempo de expiração (em milisegundos), uma requisição assíncrona faltando será movida para a lista de expedição. O padrão é **250** ms.

***fifo_expire_sync***

Este é o mesmo que o ajuste fifo_expire_async, para requisições em sincronia (leitura e gravação de O_DIRECT). O padrão é **125** ms.

***group_idle***

Quando definido, o CFQ ficará em ocioso no último processo emitindo a E/S em um cgroup. Isto deve ser definido para **1** ao usar o cgroup de E/S de peso proporcional e configurando o ***slice_idle*** to **0** (geralmente feito em armazenamento rápido).

***group_isolation***

Se a isolação de grupo estiver ativada (definida para **1**), ela fornecerá uma isolação mais forte entre grupos a custo de rendimento. Em geral, se a isolação de grupo estiver desativada, a fairness é fornecida para cargas de trabalho sequenciais somente. A ativação da isolação de grupo, proporcionará fairness para ambas cargas de trabalho aleatória e sequencial. O valor padrão é **0**) (desabilitado). Consulte o **Documentation/cgroups/blkio-controller.txt** para mais informações.

***low_latency***

Quando uma latência baixa é ativada (definida para **1**), o CFQ tenta fornecer um máximo de tempo de espera de 300 ms para cada processo emitindo E/S em um dispositivo. Isto favorece o fairness sobre o rendimento. Desabilitar a latência baixa (definindo-a para **0**) ignora a latência de alvo, permitindo que cada processo no sistema obtenha uma faixa o tempo integral. Baixa latência é ativada por padrão.

***quantum***

O quantum controla o número de E/Ss que o CFQ irá enviar ao armazenamento por vez,

principalmente limitando a profundidade da fila do dispositivo. Por padrão, isto é definido para **8**. O armazenamento pode suportar filas muito mais produndas, mas aumentar o ***quantum*** também terá um impacto negativo na latêcia, especialmente na presença de cargas de trabalho de gravação sequencial grandes.

***slice_async***

Este ajuste controla a parte de tempo alocada para cada processo que emite E/S assíncronas (gravação em buffer). Por padrão ele é ajustado para **40** ms.

***slice_idle***

Isto especifica quanto tempo o CFQ deve ficar ocioso enquanto espera por novas solicitações. O valor padrão no Red Hat Enterprise Linux 6.1 e anteriores a ele é de **8** ms. No Red Hat Enterprise Linux 6.2 e posteriores a ele, o valor padrão é **0**. O valor zero melhora a taxa de transferência de armazenamento RAID externo, removendo todos os ociosos da fila e nível de árvore de serviço. No entanto, um valor de zero pode degradar o rendimento da armazenagem não RAID interna, uma vez que aumenta o número total de procura. Para o armazenamento não-RAID, recomendamos uma ***slice_idle*** valor que é maior do que 0.

***slice_sync***

Este ajuste dita a faixa de tempo alocada para um processo emitindo E/S assíncronas (leitura ou gravação diretas). O padrão é **100** ms.

## 6.4.2. Agendador de Prazo de E/S (Deadline I/O Scheduler)

O Agendador de Prazo de E/S (Deadline I/O scheduler) tenta fornecer uma latência garantida para requisições. É importante notar que a medição de latência só começa quando os pedidos descem para o agendador de E/S (esta é uma distinção importante, pois um aplicativo pode ficar inativo esperando por descritores de requisições para ser liberado). Por padrão, as leituras são prioridade sobre gravações, uma vez que as aplicações são mais propensas a bloquear a leitura de E/S.

Prazos de reparos de E/S em lotes. Um lote é uma seqüência de E/S de leitura ou gravação, que estão em ordem crescente de LBA (o elevador de uma só mão). Após o processamento de cada lote, o agendadorde E/S verifica se os pedidos de gravação faltaram por muito tempo, e então decide se pretende iniciar um novo lote de leitura ou gravação. A lista FIFO de pedidos só é verificada para os pedidos expirados no início de cada lote, e depois para a direção desse lote de dados. Assim, se um lote de gravação é selecionado, e há uma solicitação de leitura expirada, a requisição de leitura não será atendida até que o lote de gravação seja concluído.

**Ajustáveis**

***fifo_batch***

Isto determina o número de leituras e gravações a serem emitidos em uma única vez. O padrão é **16**. Configurar este número para um valor maior resultará em melhor rendimento mas também aumentará a latência.

***front_merges***

Você pode definir este ajuste para **0** se você souber se sua carga de trabalho vai gerar alguma mesclagem de frente. A não ser que você tenha medido o cabeçalho desta verificação, recomenda-se deixá-lo como a configuração padrão. (**1**).

***read_expire***
: Este ajuste permite que você ajuste o número de milisegundos no qual a requisição de leitura deve ser atendida. Por padrão isto é definido para **500** ms (meio segundo).

***write_expire***
: Este ajuste permite que você defina o número de milisegundos no qual uma requisição de gravação deve ser atendida. Por padrão ele é definido para **5000** ms (cinco segundos).

***writes_starved***
: Este ajuste controla quantos grupos de leituras podem ser processados antes de processar um único grupo de gravação. Quanto maior o ajuste mais preferência é dada à leitura.

### 6.4.3. Noop

O agendador Noop E/S implementa algorítimos simples de First-in-first-out (FIFO). A mesclagem de requisições acontece em uma camada de bloco genérica mas é um cache simples de último toque. Se um sistema é limitado à CPU e o armazenamento é rápido, este agendador de E/S pode ser o melhor a utilizar.

Segue abaixo ajustes disponíveis para a camada de bloco.

**/sys/block/sdX/queue tunables**

**add_random**
: Em alguns casos, o cabeçalho de eventos de E/S que contribuem para o pool de entropia para **/dev/random** é mensurável. Em alguns casos, pode ser melhor ajustar este valor para 0.

***max_sectors_kb***
: Por padrão, o tamanho máximo de requisição para disco é de **512** KB. Este ajuste pode ser utilizado tanto para aumentar ou diminuir o valor. O valor mínimo é limitado por tamanho de bloco lógico; o valor máximo é limitado pelo ***max_hw_sectors_kb***. Existem alguns SSDs que possuem um pior desempenho quando os tamanhos de E/S excedem o tamanho de bloco de remoção interno. Em alguns casos recomendamos ajustar o ***max_hw_sectors_kb*** para baixo para apagar o tamanho do bloco. Você pode testar isto utilizando um gerador de E/S tal como o **iozone** ou **aio-stress**, variando o tamanho de histórico, por exemplo, de **512** bytes para **1** MB.

***nomerges***
: Este ajuste é inicialmente um assistente de depuração. A maioria de cargas de trabalhos de mesclagem de requisição (até mesmo em armazenamento mais rápido tal como SSDs). Em alguns casos, no entanto, deseja-se desabilitar a mesclagem, tal como quando você vê quantos IOPS um backend de armazenamento pode processar sem desabilitar o cabeçalho de leitura ou realizando uma E/S aleatória.

***nr_requests***
: Cada fila de requisição tem um limiete no número total de descritores de requisição que pode ser alocado para cada uma das E/Ss das leituras e gravações. Por padrão, o número é **128**, o que significa 128 leituras e 128 gravações que podem ser enfileiradas uma por vez antes de

colocar um processo em inativo. O processo inativo é o próximo a tentar alocar uma requisição, não necessariamente o processo que tenha alocado todas as requisições disponíveis.

Se você possui aplicativo de latência sensível, você deve considerar diminuir o valor do ***nr_requests*** em sua fila de requisições e limitar a profundidade da fila de comando no armazenamento para um número mais baixo (até mesmo tão baixo quanto **1**), para que a E/S writeback não possa alocar todos os descritores de requisições disponíveis e preencher a fila de dispositivo com uma E/S de gravação. Uma vez que o ***nr_requests*** tenha sido alocado, todos os outros processos que estão tentando realizar uma E/S serão transformados em inativos para esperar por requisições a ficarem disponíveis. Isto torna as coisas mais justas, pois as requisições são então distribuídas em uma moda de repetição alternada (ao invés de deixar um processo consumí-los todos em sucessão rápida). Note que este é o único problema ao utilizar a data limite ou agendadores de noop, pois a configuração CFQ padrão protege contra esta situação.

***optimal_io_size***

Em algumas circunstâncias o armazenamento adjacente irá reportar um tamanho de E/S adequado. Isto é mais comum em hardware e software RAID, onde o tamanho da E/S adequada é o tamanho da faixa. Se este valor é reportado, os aplicativos devem emitir uma E/S alinhada e em múltiplos do tamanho de E/S adequado sempre que possível.

***read_ahead_kb***

O sistema operacional pode detectar quando um aplicativo está lendo dados seqüencialmente de um arquivo ou de disco. Nesses casos, ele executa um algoritmo inteligente de leitura antecipado, em que mais dados do que é solicitado pelo usuário é lido a partir do disco. Assim, nas próximas tentativas do usuário de ler um bloco de dados, ele já o fará no cache da página do sistema operacional. A desvantagem é que o sistema operacional pode ler mais dados do disco do que o necessário, o que ocupa espaço no cache de página até que ele seja expulso por causa da alta pressão de memória. Depois de vários processos realizando leituras falsas futuras, aumentaria a pressão de memória nesta circunstância.

Para os dispositivos de mapeador, recomenda-se aumentar o valor do ***read_ahead_kb*** para um número maior, tal como **8192**. A razão é que um mapeador de dispositivo é geralmente criado de dispositivos adjacentes múltiplos. Configurar este valor para um padrão (**128** KB) multiplicado pelo número de dispositivos que você está mapeando é um bom começo para ajuste.

***rotational***

Discos rígidos tradicionais são rotacionais (feitos de discos rodopiantes). Os SSDs no entanto, não. A maioria de SSDs promoverão isto adequadamente. Se, no entanto, você encontar um dispositivo que não promova esta bandeira adequadamente, pode ser necessário configurar o rotacional para **0** manualmente; quando o rotacional estiver desabilitado, o elevador de E/S não usará lógica, que significa reduzir buscas, uma vez que existe pouca penalidade para busca de operações em mídia não rotacional.

***rq_affinity***

Conclusões de E/S podem ser processadas em uma CPU diferente daquela que emitiu a E/S. Configurar o ***rq_affinity*** para **1** faz com que o kernel entregue conclusões para a CPU na qual a E/S foi emitida. Isto pode aprimorar a efetividade do cache de dados de CPU.

# Capítulo 7. Sistemas de Arquivos

Leia este capítulo para obter uma visão geral dos sistemas de arquivos suportados para uso com o Red Hat Enterprise Linux, e como otimizar seus desempenhos.

## 7.1. Considerações de ajustes para Sistemas de Arquivo

Existem diversas considerações de ajustes comuns a todos os sistemas de arquivo: formatar e montar opções selecionadas em seu sistema, e ações disponíveis para aplicativos que podem aprimorar seu desempenho em um sistema específico.

### 7.1.1. Formatando Opções

#### Tamanho de bloco de sistema de arquivo

O tamanho do bloco pode ser selecionado no tempo **mkfs**. A classe de tamanhos válidos depende do sistema: o limite acima é o tamanho máximo da página do sistema host, equanto o limite mais baixo depende do sistema de arquivo utilizado. O tamanho do bloco padrão é adequado para a maioria dos casos de uso.

Se você espera criar muitos arquivos menores do que o tamanho do bloco padrão, você pode definir o tamanho do bloco menor para minimizar a quantia de espaço disperdiçada no disco. Note que no entanto, ao configurar um tamanho de bloco menor você poderá limitar o tamanho máximo do sistema de arquivo, e pode causar cabeçalho de tempo de execução adicional, especialmente para arquivos maiores do que o tamanho de bloco selecionado.

#### Geometria de Sistema de Arquivo

Se seu sistema utiliza armazenamento em faixas tal qual o RAID, você poderá aprimorar o desempenho, alinhando os dados e metadados com a geometria de armazenamento adjacente no tempo **mkfs**. Para o RAID software (LVM ou MD) e alguns armazenamentos de hardware corporativos, esta informação é enfileirada e definida automaticamente, mas em muitos casos o administrador precisa especificar esta geometria manualmente com o **mkfs** na linha de comando.

Consulte o *Storage Administration Guide* para mais informações sobre criar e manter estes sistemas de arquivo.

#### Diários externos

Cargas de trabalho de metadados intensivos significa que a seção de log de um sistema de arquivo de agendamento (como o ext4 e XFS) é atualizado muito frequentemente. Para minimizar o tempo de busca do sistema de arquivos do diário, você pode colocar o diário no armazenamento dedicado. Note, no entanto, que a colocação do diário em armazenamento externo é mais lenta que o sistema de arquivos primário possa anular qualquer vantagem potencial associado com o uso de armazenamento externo.

**Atenção**

Certifique-se de que o diário externo é confiável. A perda de um dispositivo de diário externo causará danos no sistema de arquivo.

Diários externos são criados no tempo **mkfs**, com os dispositivos de diário sendo especificados no tempo de montagem. Consulte as páginas man **mke2fs(8)**, **mkfs.xfs(8)**, e **mount(8)** para maiores informações.

### 7.1.2. Opções de montagem

#### Barreiras

Uma barreira de gravação é um mecanismo do kernel usado para assegurar que os metadados do sistema de arquivos foi gravado corretamente e ordenado em armazenamento persistente, mesmo quando os dispositivos de armazenamento com gravações de caches voláteis perdem o poder. Os sistemas de arquivos com barreiras de gravação ativadas também garantem que todos os dados transmitidos via **`fsync ()`** persistem através de uma queda de energia. O Red Hat Enterprise Linux permite barreiras por padrão em todos os hardwares que as suportam.

No entanto, permitir que as barreiras de gravação diminui significativamente algumas aplicações, especificamente, aplicativos que usam **`fsync ()`** pesadamente, ou que cria e apaga muitos arquivos pequenos. Para o armazenamento, sem o cache de gravação volátil, ou no caso raro onde as inconsistências do sistema de arquivos e perda de dados após uma perda de potência é aceitável, barreiras podem ser desativadas usando a opção de montagem ***`nobarrier`***. Para mais informações, consulte o *Guia de Administração de Armazenamento* .

#### Tempo de Acesso (noatime)

HIstóricamente, quando um arquivo é lido, o tempo de acesso (***`atime`***) para aquele arquivo deve ser atualizado no metadado inode, que involve E/S de gravações adicionais. Se os metadados forem precisos ***`atime`*** não forem necessários, monte o sistema de arquivo com a opção ***`noatime`*** para eliminar estas atualizações de metadados. Na maioria dos casos, no entanto, o ***`atime`*** não é um cabeçalho grande devido ao comportamento do atime relativo padrão (ou ***`relatime`***) no kernel do Red Hat Enterprise Linux 6. O comportamento do ***`relatime`*** atualiza somente o ***`atime`*** se o ***`atime`*** for mais vejo do que o tempo de modificação (***`mtime`***) ou tempo de mudança de status (***`ctime`***).

**Nota**

Habilitar a opção ***`noatime`*** também habilita o comportamento do ***`nodiratime`***; não há necessidade de definir ambos ***`noatime`***e ***`nodiratime`***.

#### Aumento de suporte do read-ahead (leitura antecipada)

O Read-ahead acelera acesso de arquivo buscando antecipadamente dados e carregando-os no cache de página para que possa estar disponível antes na memória ao invés de vir do disco. Algumas cargas de trabalho, tais como aquelas que envolvem transmissão contínua pesada de E/S sequencial, se beneficiam de valores altos de read-ahead.

A ferramenta **`tuned`** e o uso da faixa de LVM eleva o valor da leitura antecipada, mas isso nem sempre é suficiente para algumas cargas de trabalho. Além disso, o Red Hat Enterprise Linux nem sempre é capaz de definir um valor de leitura antecipada correto baseado no que ele pode detectar do seu sistema de arquivos. Por exemplo, se uma matriz de armazenamento poderosa se apresenta para o Red Hat Enterprise Linux como um único LUN potente, o sistema operacional não irá tratá-lo como uma matriz LUN potente, e, portanto por padrão, não fará pleno uso das vantagens de leitura antecipada disponíveis para o armazenamento.

Use o comando **`blockdev`** para visualizar e editar o valor read-ahead. Para visualizar o valor read-ahead atual para um dispositivo de bloco particular, execute:

```
# blockdev -getra device
```

Para modificar o valor read-ahead para aquele dispositivo de bloco, execute o comando a seguir. ***N*** representa o número de setores de 512 bites.

```
# blockdev -setra N device
```

Note que o valor selecionado com o comando **blockdev** não persistirá entre as inicializações. Nós recomendados criar um nível de execução do script **init.d** para definir este valor durante a inicialização.

### 7.1.3. Manutenção de sistema de arquivo.

#### Descartar blocos não utilizados

Operações de descarte em Lote e online são recursos de sistemas de arquivos montados que descartam blocos que não estão sendo utilizados pelo sistema de arquivo. Estas operações são úteis para ambos drives de estado sólido e armazenamento finalmente provisionado.

*Operações de Discard em Lote* são executadas pelo usuário com o comando **fstrim**. Este comando descarta todos os blocos não usados em um sistema de arquivo que coincida com os critérios de usuário. Ambos os tipos de operação são suportados para uso com o XFS e os sistemas de arquivo ext4 no Red Hat Enterprise Linux 6.2 e posteriores, desde que o dispositivo de bloco adjacente ao sistema de arquivo suporte as operações de discard físicas. As operações de Discard Físico são suportadas se o valor de **/sys/block/*device*/queue/discard_max_bytes** não for zero.

O *Operações de Discard online* são especificadas no tempo de montagem com a opção **-o discard** (tanto no **/etc/fstab**quanto como parte do comando **mount**), e executados em tempo real sem a intervenção do usuário. As operações de discard somente descartam blocos que esteja transitando de um usado para um livre. As operações discard são suportadas nos sistemas de arquivo ext4 no Red Hat Enterprise Linux 6.2 e posteriores, e nos sistemas de arquivo XFS no Red Hat Enterprise Linux 6.4 e posteriores.

A Red Hat recomenda as operações de discard em lote a não ser que a carga de trabalho do sistema seja tanta que o discard em lote não seja possível, ou se as operações de discard online forem necessárias para manter o desempenho.

### 7.1.4. Considerações de Aplicativos

#### Pré-alocação

Os sistemas de arquivo ext4, XFS, e GFS2 apoiam a pré-alocação eficiente do espaço através da chamada glibc **fallocate (2)**. Nos casos em que os arquivos possam tornar-se muito fragmentados devido aos padrões de gravação, levando a má performance de leitura, a pré-alocação de espaço pode ser uma técnica útil. Pré-alocação marca de espaço em disco como se tivesse sido alocado para um arquivo, sem gravar nenhum dado naquele espaço. Até que os dados reais sejam gravados em um bloco pré-alocado, as operações de leitura retornarão como zeros.

## 7.2. Perfis para desempenho de sistema de arquivo.

A ferramenta **tuned-adm** permite que usuários troquem facilmente entre um número de perfis que foram criados para melhorar desempenho para casos de uso específico. Os perfis que são especialmente úteis em aprimorar o desempenho do armazenamento são:

***desempenho de latência***
: Um perfil de servidor para latência típica de ajuste de desempenho. Ele desabilita os mecanismos de economia de energia do **tuned** e **ktune**. Os modos **cpuspeed** mudam para **performance**. O elevador de E/S é modificado para **deadline** para cada dispositivo. O parâmetro ***cpu_dma_latency*** é registrado com um valor de **0** (a latência menor possível) para qualidade de serviço de gerenciamento de energia para limitar a latência onde for possível.

***desempenho de rendimento***
: Um perfil de servidor para um ajuste de desempenho de rendimento típico. Este perfil é recomendado se o sistema não tiver o armazenamento de classe corporativa. É o mesmo que **latency-performance**, exceto:

  - ***kernel.sched_min_granularity_ns*** (granularidade mínima de agendador) é definida para **10** milisegundos,
  - ***kernel.sched_wakeup_granularity_ns*** (granularidade de ativação de agendador) é definida para **15** milisegundos,
  - ***vm.dirty_ratio*** (taxa suja de máquina virtual) é definida para 40%, e
  - páginas transparentes enormes são habilitadas.

***enterprise-storage***
: Este perfil é recomendado para configurações de servidor de tamanho corporativo com armazenamento de classe corporativa, incluindo proteção e gerenciamento de cache de controlador com backup de bateria de um cache em disco. É o mesmo que o perfil **desempenho de rendimento**, exceto:

  - o valor ***readahead*** é definido para **4x**, e
  - sistemas de arquivo non root/boot são remontados com **barrier=0**.

Mais informações sobre **tuned-adm**está disponível na página man (**man tuned-adm**),ou em *Guia de Gerenciamento de Energia* disponível a partir do link http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

## 7.3. Sistemas de Arquivos

### 7.3.1. Sistema de Arquivo Ext4

O sistema de arquivo ext4 é uma extensão escalável do sistema de arquivo ext3 padrão disponível noRed Hat Enterprise Linux 5. O Ext4 é agora o sistema de arquivo padrão para Red Hat Enterprise Linux 6 e é suportado por um tamanho máximo de arquivo do sistema de 16TB, e um único tamanho máximo de 16TB.Isto também remove o limite de sub-diretório de 32000 presente no ext3.

**Nota**

Para sistemas de arquivo maiores do que 16TB, recomendamos o uso do sistema de arquivo de alta capacidade escalável tal como XFS. Para mais informações, veja Seção 7.3.2, “O Sistema de Arquivo XFS”.

O padrão do sistema de arquivo ext4 é adequado para a maioria das cargas de trabalho, mas se a análise de desempenho mostra que o comportamento de sistema de arquivo está impactando o desempenho, diversas opções de ajuste estão disponíveis:

### Inicialização de tabela do Inode

Para sistemas de arquivo muito grandes, o processo **mkfs.ext4** pode levar um longo tempo para inicializar todas as tabelas de inodes no sistema de arquivo. Este processo pode ser deferido com a opção ***-E lazy_itable_init=1***. Se for utilizado, os processos do kernel continuarão a inicializar o sistema de arquivo após ser montado. A taxa na qual esta inicialização acontece pode ser controlada com a opção ***-o init_itable=n*** para o comando **mount**, onde a quantia de tempo gasta na realização n esta inicialização de fundo é mais ou menos 1/n. O valor padrão para **n** é de **10**.

### Comportamento auto-fsync

Como alguns aplicativos nem sempre realizam o **fsync()** adequadamente após renomear um arquivo existente, ou truncado e regravado, o padrão do ext4 é a sincronização automática de arquivos após operações de substituição via renome e substituição via truncar. Este comportamento é altamente consistente com comportamentos de sistema de arquivo ext3 mais velhos. No entanto, as operações de **fsync()** podem consumir bastante tempo, portanto se este comportamento automático não for necessário, use a opção ***-o noauto_da_alloc*** com o comando **mount** para desabilitá-lo Isto significa que o aplicativo deverá utilizar explicitamente o **fsync()** para assegurar persistência de dados.

### Prioridade de E/S de Diário

Por padrão, a E/S de comprometimento de diário recebe uma prioridade uma pouco mais alta do que uma E/S normal. Esta prioridade pode ser controlada com a opção ***journal_ioprio=n*** do comando **mount**. O valor padrão é **3**. Valores válidos variam desde 0 à 7, sendo 0 a E/S com prioridade mais alta.

Para outras opções **mkfs** e ajuste, veja as páginas man **mkfs.ext4(8)** e **mount(8)** assim como o arquivo **Documentation/filesystems/ext4.txt**no pacote *kernel-doc*.

## 7.3.2. O Sistema de Arquivo XFS

XFS é um sistema de arquivo de 64 bits, altamente escalável e robusto. Ele é baseado inteiramente em extensão, portanto ele suporta tamanhos bem grandes de arquivos e sistemas de arquivos. O número de arquivos que um sistema XFS pode suportar é limitado somente à disponibilidade de espaço no sistema de arquivo.

O XFS suporta agendamento de metadados, o qual facilita a recuperação de travamento rápida. Os sistemas de arquivo XFS também podem ser desfragmentados e expandidos enquanto são montados e ativados. Além disso, o Red Hat Enterprise Linux 6 suporta o backup e recperações de utlitários específicos ao XFS.

O XFS usa a alocação baseada em extensão, e apresenta um número de esquemas de alocação tais como alocação atrasada e pré-alocação explícita. A alocação baseada em extensão fornece um método de rastreamento de espaço utilizado em um sistema de arquivo mais compacto e eficiente, e aprimora o desempenho de arquivos grandes reduzindo a fragmentação e o espaço consumido pelo metadado. A alocação atrasada aprimora a chance que um arquivo terá de ser gravado em um grupo contíguo de blocos, reduzindo a fragmentação e aprimorando o desempenho. A pré-alocação pode ser usada para prevenir fragmentação de um modo geral em casos onde o aplicativo sabe a quantidade de dados que precisa para gravar antecipadamente.

O XFS fornece escalabilidade de E/S excelente, utilizando b-trees para indexar todos os dados de usuários e metadados. As contagens de objetos crescem a medida que todas as operações em índices

herdam a escalabilidade de logarítimos de b-trees adjacentes. Algumas opções de ajuste de XFS em tempo **mkfs** variam em profundidade de b-trees, que modificam as características de escalabilidade de subsistemas diferentes.

#### 7.3.2.1. Ajuste básico para XFS

Em geral, o formato do XFS padrão e opções de montagem para a maioria de cargas de trabalho são adequadas; a Red Hat recomenda que os valores padrão sejam utilizados a não ser em mudanças de configuração especificas esperadas para beneficiar a carga de trabalho do sistema de arquivo. Se o software RAID estiver em uso, o comando **mkfs.xfs** configura automaticamente a si próprio com a unidade de faixa correta e profundidade para alinhar com o hardware. Isto pode precisar ser configurado manualmente se o hardware RAID estiver em uso.

A opção de montagem **inode64** é altamente recomendada para sistemas de arquivo em muli-terabyte, exceto onde o sistema de aquivo for exportado via NFS e clientes NFS de 32 bits de legacia, que requerem acesso ao sistema de arquivo.

A opção de montagem **logbsize** é recomendada para sistemas de arquivo que são modificados frequentemente, ou em intermitências. O valor padrão é de **MAX** (32 KB, unidade de faixa de log), e o tamanho máximo é 256 KB. O valor de 256 KB é recomendado para sistemas de arquivo que passam por modificações pesadas.

#### 7.3.2.2. Ajuste avançado para XFS

Antes de mudar os parâmetros de XFS, você precisará entender porque os parâmetros de XFS padrão estão causando problemas de desempenho. Isto envolve compreensão do que seu aplicativo está fazendo, e como o sistema de arquivo está reagindo àquelas operações.

Os problemas de desempenho observáveis que podem ser corrigidos ou reduzidos através do ajuste são geralmente causados pela fragmentação de arquivo ou contenção de recursos no sistema de arquivo. Existem formas diferentes de endereçar estes problemas e em alguns casos reparar o problema irá requerer que o aplicativo, ao invés da configuração de sistema de arquivo, seja modificado.

Se você não passou por este processo anteriormente, recomenda-se que você entre em contato com o engenheiro de suporte da Red Hat para obter assistência.

##### Otimização para um número grande de arquivos.

O XFS impõe um limite arbitrário para o número de arquivos que um sistema de arquivos pode conter. Em geral, esse limite é alto o suficiente para que ele nunca seja atingido. Se você sabe que o limite padrão será insuficiente antes do tempo, você pode aumentar a percentagem de espaço do sistema de arquivo permitido para inodes com os **mkfs.xfs**. Se você encontrar o limite do arquivo após a criação do sistema de arquivos (normalmente indicado por erros ENOSPC ao tentar criar um arquivo ou pasta, apesar de espaço livre disponível), você poderá ajustar o limite com os **xfs_growfs** .

##### Otimização para um grande número de arquivos em um diretório único

O tamanho do bloco do diretório é fixado para a vida de um sistema de arquivos, e não pode ser alterado, salvo mediante a formatação inicial com **mkfs**. O bloco mínimo de diretório é o tamanho do bloco do sistema de arquivos, o qual tem como padrão **MAX** (4 KB, o tamanho do arquivo de bloco do sistema). De um modo geral, não há nenhuma razão para reduzir o tamanho do bloco de diretório.

Como a estrutura de diretório é baseado b-tree, mudar o tamanho do bloco afeta a quantidade de informações de diretório que podem ser recuperadas ou modificados por E/S física. Quanto maior o diretório se torna, mais E/S cada operação irá requerer em um determinado tamanho do bloco.

No entanto, quando maiores tamanhos de bloco de diretório estão em uso, mais CPU é consumida por

cada operação de modificação em relação à mesma operação em um sistema de arquivos com um diretório de menor tamanho do bloco. Isto significa que para tamanhos pequenos de diretório, blocos grandes de diretórios irão resultar em menor desempenho modificação. Quando o diretório atinge um tamanho onde E/S é o fator limitante do desempenho, tamanhos grandes de diretórios de bloco possuem um melhor desempenho.

A configuração padrão de um tamanho de bloco de sistema de aquivo de 4 KB e um tamanho de bloco de diretório de 4 KB é o melhor para diretórios com até 1-2 milhões de entradas com um comprimento de nome de bytes 20-40 por entrada. Se o seu sistema de arquivos requer mais entradas, maiores tamanhos de bloco de diretório tendem a ter um melhor desempenho - um tamanho de bloco 16 KB é melhor para sistemas de arquivos com 1-10.000.000 entradas de diretório e um tamanho de bloco 64 KB é melhor para sistemas de arquivos com mais de 10 milhões de entradas de diretório.

Se a carga de trabalho usa pesquisas de diretório aleatórias mais do que modificações (ou seja, as leituras de diretório são muito mais comuns e importantes do que as gravações de diretório), então os limiares acima para aumentar o tamanho do bloco é de aproximadamente uma ordem de magnitude menor.

### Otimização para simultaneidade

Ao contrário de outros sistemas de arquivos, o XFS pode realizar vários tipos de operações de alocação e desalocação simultaneamente, desde que as operações estejam ocorrendo em objetos não-compartilhados. Alocação ou desalocação de extensões podem ocorrer simultaneamente, desde que as operações simultâneas ocorram em diferentes grupos de alocação. Da mesma forma, a alocação ou desalocação de inodes podem ocorrer simultaneamente, desde que as operações simultâneas afetem diferentes grupos de alocação.

O número de grupos de alocação torna-se importante quando a utilização de máquinas com uma elevada contagem de CPU e aplicativos multi-threaded que tentam executar operações simultaneamente. Se existir apenas quatro grupos de atribuição, então operações de metadados paralelas e sustentadas serão dimensionadas somente até aquelas quatro CPUs (o limite de simultaneidade fornecida pelo sistema). Para os sistemas de arquivos pequenos, assegure que o número de grupos de atribuição é suportado pela simultaneidade fornecida pelo sistema. Para sistemas de arquivos grandes (dezenas de terabytes e maiores) as opções de formatação padrão geralmente criam grupos de alocação suficientes para não limitar a simultaneidade.

Os aplicativos devem estar cientes de pontos únicos de contenção, a fim de usar o paralelismo inerente à estrutura do sistema de arquivos XFS. Não é possível modificar um diretório ao mesmo tempo, portanto os aplicativos que criam e removem grandes quantidades de arquivos devem evitar o armazenamento de todos os arquivos em um único diretório. Cada diretório criado é colocado em um grupo de alocação diferente, portanto, técnicas, tais como arquivos de hash sobre múltiplos sub-diretórios fornecem um padrão de armazenamento escalável em relação ao uso de um único diretório grande.

### Otimização para aplicativos que utilizam atributos estendidos.

XFS pode armazenar pequenos atributos diretamente no inode se houver espaço disponível. Se o atributo encaixa no inodo, então ele pode ser recuperado e modificado sem a necessidade de E/S adicionais para recuperar blocos de atributos distintos. O diferencial de desempenho entre os in-line e atributos out-of-line podem facilmente ser uma ordem de magnitude mais lenta para os atributos de out-of-line.

Para o tamanho do inode padrão de 256 bytes, cerca de 100 bytes de atributo de espaço está disponível, dependendo do número de apontadores de extensão de dados também armazenados no inode. O tamanho de inodo padrão é realmente útil apenas para armazenar um número pequeno de

pequenas atributos.

Aumentando o tamanho do inode em tempo mkfs pode aumentar a quantia de espaço disponível para armazenar atributos in-line. Um tamanho de inode de 512 bytes aumenta o espaço disponível para cerca de 350 bytes. Um inode de 2 KB possui cerca de 1900 bytes de espaço disponível.

Existe, no entanto, um limite para o tamanho dos atributos individuais que podem ser armazenados in-line - existe um limite máximo de tamanho de 254 bytes para o nome de atributo e o valor (isto é, um atributo com um comprimento de nome de 254 bytes e um comprimento de 254 bytes valor ficará in-line). Exceder esses limites de tamanho obriga os atributos de linha, mesmo que não tenha espaço suficiente para armazenar todos os atributos do inode.

### Otimização para modificações de metadados sustentados.

O tamanho do log é o principal fator na determinação do nível possível de modificação de metadados sustentados. O dispositivo de log é circular, portanto, antes da parte final poder ser sobrescrita, todas as modificações no registro devem ser gravadas no local real no disco. Isto pode envolver uma quantidade significativa de procura para reproduzir todos os metadados sujos. A configuração padrão escala o tamanho do log em relação ao tamanho do sistema de arquivos em geral, assim, na maioria dos casos o tamanho do log não vai precisar de ajuste.

Um dispositivo de log pequeno irá resultar em write-back de metadados muito freqüente - o registro será constantemente empurrado em sua parte final para liberar espaço e assim metadados freqüentemente modificados serão freqüentemente gravados no disco, fazendo com que as operações sejam lentas.

Aumentar o tamanho do log aumenta o período de tempo entre eventos que empurram sua parte final. Isso permite uma melhor agregação de metadados sujos, resultando em melhores padrões de write-back de metadados, e menos de write-back de metadados frequentemente alterados. O compromisso é que os logs maiores requeiram mais memória para acompanhar todas as mudanças pendentes na memória.

Se você tem uma máquina com memória limitada, então logs grandes não são benéficos porque restrições de memória causarão write-back de metadados muito antes de obter benefícios de um grande log. Nestes casos, os registros menores, em vez de maiores, muitas vezes, fornecem um melhor desempenho porque write-back de metadados a partir do registro a falta de espaço é mais eficiente que o write-back impulsionado pela recuperação da memória.

Você deveria tentar sempre alinhar o log com a unidade de faixa adjacente que contenha o sistema de arquivo. O **`mkfs`** faz isto por padrão para os dispositivos MD e DM, mas para o hardware RAID é possível que precise ser especificado. Configurá-lo corretamente evita todas as possibilidades do log de E/S causar uma E/S desalinhada e operações subsequentes de leitura-modificar-gravação ao gravar as modificações em disco.

Operação de log pode ser melhorada através da edição de opções de montagem. Aumentar o tamanho dos buffers de log na memória (**`ogbsize`**) aumenta a velocidade com que as mudanças podem ser gravadas no log. O tamanho do buffer de log padrão é **`MAX`** (32 KB, unidade de faixa de log), e o tamanho máximo é de 256 KB. Em geral, um maior valor resulta em desempenho mais rápido. No entanto, sob cargas de trabalho fsync-pesadas, buffers de log pequenos podem ser visivelmente mais rápidos do que os grandes buffers grandes com um grande alinhamento da unidade de faixa.

A opção de montagem **`delaylog`** também melhora o desempenho de modificação de metadados sustentados, reduzindo o número de alterações no log. Ela consegue isso através da agregação de mudanças individuais na memória antes de gravá-los no log: metadados modificados freqüentemente é gravado no log periodicamente em vez de em cada modificação. Essa opção aumenta o uso de memória

de rastreamento de metadados sujos e aumenta as operações de perdas potenciais quando ocorre um travamento, mas pode melhorar a velocidade de modificação de metadados e escalabilidade por uma ordem de magnitude ou mais. O uso desta opção não reduz a dados ou a integridade de metadados quando **fsync**, **fdatasync** ou **sync** são usados para garantir que os dados e metadados sejam gravados no disco.

## 7.4. Clustering

Armazenamento em cluster fornece uma imagem do sistema de arquivos consistente em todos os servidores em um cluster, permitindo que os servidores leiam e gravem em um único sistema de arquivos compartilhado. Isso simplifica a administração de armazenamento, limitando tarefas como instalação de reparos e aplicativos em um sistema de arquivos. Um sistema de arquivos em todo o cluster também elimina a necessidade de cópias redundantes de dados de aplicativos, simplificando o backup e a recuperação de desastres.

Red Hat's High Availability Add-On fornece armazenamento em cluster em conjunto com o Red Hat Global File System 2 (parte do Resilient Storage Add-On).

### 7.4.1. Global File System 2

Global File System 2 (GFS2) é um sistema de arquivos nativos que interage diretamente com o sistema de arquivos do kernel Linux. Ele permite que vários computadores (nós) compartilhem simultaneamente o mesmo dispositivo de armazenamento em cluster. O sistema de arquivos GFS2 é em grande parte auto ajustável, mas o ajuste manual é possível. Esta seção descreve as considerações de desempenho ao tentar ajustar o desempenho manualmente.

A Red Hat Enterprise Linux 6.4 apresenta melhorias no gerenciamento de fragmentação de arquivo no GFS2. Os arquivos criados pelo Red Hat Enterprise Linus 6.3 ou anteriores tinham a tendência à fragmentação de arquivo se múltiplos arquivos fossem gravados ao mesmo tempo por mais de um processo. Esta fragmentação fez com que tudo ficasse mais lento, especialmente em cargas de trabalhos envolvendo grandes arquivos. Com o Red Hat Enterprise Linux 6.4, as gravações simultâneas resultam em menos fragmentações de arquivos e portanto em melhor desempenho para estas cargas de trabalho.

Embora não exista uma ferramenta de desfragmentação para GFS2 no Red Hat Enterprise Linux, você pode desfragmentar arquivos individuais, identificando-os com a ferramenta **filefrag**, copiá-los para arquivos temporários, e renomear os arquivos temporários para substituir os originais. (Este procedimento também pode ser feito em versões anteriores a 6.4, enquanto que a gravação é feita sequencialmente.)

Como o GFS2 usa um mecanismo de bloqueio global que potencialmente requer a comunicação entre os nós de um cluster, o melhor desempenho será alcançado quando o sistema é projetado para evitar uma contenção de arquivo e diretório entre esses nós. Alguns métodos para evitar contenção são os seguintes:

- Arquivos pré-alocados e diretórios com o **fallocate** onde possível, para otimizar o processo de alocação e evitar a necessidade de bloquear páginas fonte.
- Minimizar as áreas do sistema de arquivos que são compartilhadas entre vários nós para minimizar a invalidação do cache cross-nó e melhorar o desempenho. Por exemplo, se vários nós montarem o mesmo sistema de arquivos, mas acessarem diferentes sub-diretórios, você provavelmente vai conseguir um melhor desempenho movendo um subdiretório para um sistema de arquivo separado.
- Escolha um tamanho de grupo de recursos ideal e número. Isso depende de tamanhos de arquivo típicos e espaço livre disponível no sistema, e afeta a probabilidade de que vários nós tentarão usar um grupo de recursos simultaneamente. Muitos grupos de recursos podem retardar a alocação de

blocos, enquanto o espaço de alocação é localizado, enquanto muito poucos grupos de recursos podem causar contenção de bloqueio durante a desalocação. Em geral, é melhor testar várias configurações para determinar o que é melhor para a sua carga de trabalho.

No entanto, a contenção não é o único problema que pode afetar o desempenho de sistema de arquivo GFS2. Outras práticas para aprimorar o desempenho geral são:

- Selecione seu hardware de armazenamento de acordo com os modelos de E/S esperados dos nós de cluster e os requerimentos de desempenho do sistema de arquivo.
- Use armazenamento de estado sólido onde possível para diminuir tempo de busca.
- Crie um sistema de arquivos de tamanho apropriado para o seu trabalho, e assegure-se que o sistema de arquivos nunca está em mais de 80% da capacidade. Sistemas de arquivos menores terão o tempo de backup proporcionalmente mais curtos, e requerem menos tempo e memória para o controle do sistema de arquivos, mas estão sujeitos a elevada fragmentação caso sejam pequenos demais para a sua carga de trabalho.
- Defina tamanhos de diários maiores para cargas de trabalho de metadados intensivo, ou quando dados com diário estiver em uso. Embora este use mais memória, ele melhora o desempenho, pois mais espaço diário está disponível para armazenar dados antes de uma gravação ser necessária.
- Assegure-se de que o relógico nos nós de GFS2 estão sincronizados para evitar problemas com os aplicativos em rede. Recomendamos o uso do NTP (Network Time Protocol).
- A menos que os tempos de acesso de arquivo ou diretório sejam críticos para a operação de seu aplicativo, monte o sistema de arquivo com as opções de montagem **`noatime`** e **`nodiratime`**.

**Nota**

Red Hat recomenda o uso da opção **`noatime`** com o GFS2.

- Se você precisar usar quotas, tente reduzir a freqüência das operações de sincronização de cota ou usar a sincronização de quota difusa para evitar problemas de desempenho decorrentes de atualizações de arquivos de quotas constantes.

**Nota**

A conta de cotas "difusas" (fuzzy) pode permitir que usuários ou grupos excedam um pouco do limite de cota. Para minimizar isto, o GFS2 reduz o período de sincronização, de forma dinâmica quando um usuário ou grupo se aproxima do limite de cota.

Para mais informações sobre os aspectos do ajuste de desempenho do GFS2 consulte o *Global File System 2* guide, available from http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

# Capítulo 8. Networking

Com o tempo, pilha de rede do Red Hat Enterprise Linux foi atualizada com inúmeros recursos de otimização automatizadas. Para a maioria das cargas de trabalho, as configurações de rede auto-configurados proporcionam desempenho otimizado.

Na maioria dos casos, problemas de desempenho de rede são realmente causados por um defeito no hardware ou infra-estrutura deficiente. Tais causas estão além do escopo deste documento, os problemas de desempenho e soluções discutidas neste capítulo são úteis na otimização de sistemas perfeitamente funcionais.

Networking é um subsistema delicado, contendo partes diferentes com conexões sensíveis. É por isso que a comunidade open source e Red Hat investem muito trabalho na implementação de formas de otimizam automaticamente o desempenho da rede. Como tal, dado a maioria das cargas de trabalho, você pode nunca precisar reconfigurar a rede para o desempenho.

## 8.1. Melhorias de Desempenho de Rede

Red Hat Enterprise Linux 6.1 fornece as seguintes melhorias de desempenho de rede:

### Receive Packet Steering (RPS)

RPS permite que uma única fila de NIC **`rx`** tenha sua recepção de carga de trabalho **`softirq`** distribuída entre diversas CPUs. Isto ajuda a prevenir tráfego em rede de ser afunilado em uma única fila de hardware NIC.

Para permitir um RPS, especificamente os nomes de CPU de alvo em **`/sys/class/net/`*`ethX`*`/queues/`*`rx-N`*`/rps_cpus`**, substituindo ***`ethX`*** pelo nome de dispositivo correspondente do NIC (por exemplo, **`eth1`**, **`eth2`**) e ***`rx-N`***pela fila de recepção do NIC especificada. Isto permitirá que as CPUs especificadas no arquivo processem dados de uma fila ***`rx-N`*** em ***`ethX`***. Ao especificar CPUs, consider o *cache affinity* da fila [4].

### Receive Flow Steering

RFS é uma extensão do RPS, permitindo que o administrador configure uma tabela hash que é preenchido automaticamente quando receber aplicações de dados e são interrogados pela pilha de rede. Isso determina quais aplicativos estão recebendo cada pedaço de dados de rede (com base na fonte: informações sobre a rede de destino).

Com o uso desta informação, a pilha de rede pode agendar a CPU ideal para receber cada pacote. Para configurar RFS, use os seguintes ajustáveis:

**`/proc/sys/net/core/rps_sock_flow_entries`**

Isto controla o número máximo de soquetes/fluxos que o kernel pode conduzir em direção a qualquer CPU específica. Isto é um limite compartilhado com todo o sistema.

**`/sys/class/net/`*`ethX`*`/queues/`*`rx-N`*`/rps_flow_cnt`**

Isto controla o número máximo de soquetes/fluxos que o kernel pode conduzir em direção a qualquer CPU específica. (***`rx-N`***) em um NIC (***`ethX`***). Note que o resumo de todos os valores por fila para este ajustável em todos os NICs deve ser igual ou menor do que **`/proc/sys/net/core/rps_sock_flow_entries`**.

Ao contrário de RPS, o RFS permite tanto a fila de recebimento e a aplicação de compartilhar a mesma CPU durante o processamento de fluxos de pacotes. Isto pode resultar em melhor desempenho em alguns casos. No entanto, tais melhorias são dependentes de fatores como a hierarquia de cache, a carga de aplicação, e semelhantes.

### suporte de getsockopt para thin-streams do TCP

*Thin-stream* é um termo usado para caracterizar protocolos de transportes onde os aplicativos enviam dados com taxas tão baixas que os mecanismos de retransmissão de protocolo não ficam totalmente saturados. Os aplicativos que utilizam os protocolos thin-stream geralmente transportam via protocolos confiáveis como o TCP; na maioria dos casos, tais aplicativos fornecem serviços sensíveis ao tempo (por exemplo, stock trading, online gaming, sistemas de controle).

Para serviços sensíveis ao tempo, a perda de pacote pode ser devastadora para a qualidade de serviço. Para ajudar a prevenir isto, a chamada **`getsockopt`** foi aprimorada para suportar duas opções extras:

**TCP_THIN_DUPACK**
: Este Booleano habilita o disparo dinâmico de retransmissão após um dupACK para thin streams.

**TCP_THIN_LINEAR_TIMEOUTS**
: Este Booleano habilita o disparo dinâmico de limite de tempo linear para thin streams.

Ambas opções são ativadas especificamente pelo aplicativo. Para mais informações sobre estas opções, consulte o **`file:///usr/share/doc/kernel-doc-`*`versão`*`/Documentation/networking/ip-sysctl.txt`**. Para mais informações sobre o thin-streams, consulte o **`file:///usr/share/doc/kernel-doc-`*`versão`*`/Documentation/networking/tcp-thin.txt`**.

### Suporte Transparent Proxy (TProxy)

O kernel pode agora lidar com bound não local IPv4 TCP e soquetes UDP para suportar proxies transparentes. Para habilitar isto, você terá de configurar iptables adequadamente. Você também irá precisar habilitar e configurar o roteamento de política adequadamente.

Para mais informações sobre os proxies transparentes, consulte o **`file:///usr/share/doc/kernel-doc-`*`version`*`/Documentation/networking/tproxy.txt`**.

## 8.2. Configurações de Rede Otimizadas

Ajuste de desempenho é normalmente feito de forma preventiva. Muitas vezes, nós ajustamos variáveis conhecidas antes de executar um aplicativo ou implantação de um sistema. Se o ajuste revela-se ineficaz, tentamos ajustar outras variáveis. A lógica por trás desse pensamento é que *por padrão*, o sistema não está operando em um nível ideal de desempenho e, como tal, *pensamos* que é preciso ajustar o sistema de acordo. Em alguns casos, podemos fazê-lo através de palpites calculados.

Como mencionado anteriormente, a pilha de rede é auto-otimizável. Além disso, ajustar a rede de maneira efetiva requer um conhecimento profundo não apenas de como a pilha de rede funciona, mas também as necessidades de recursos de rede do sistema específico. Configuração de desempenho de rede incorreta pode realmente levar a degradação do desempenho.

Por exemplo, considere o *problema bufferfloat*. O aumento da profundidade da fila do buffer resulta em conexões TCP que têm janelas de congestionamento maior do que o link iria permitir (devido ao buffer profundo). No entanto, essas conexões também têm valores RTT enormes, pois as estruturas gastam tanto tempo em fila. Isto, por sua vez, realmente resulta em produção sub-óptima, uma vez que seria impossível detectar os congestionamentos.

Quando se trata de desempenho da rede, é aconselhável manter as configurações padrão *a menos que* um problema de desempenho particular, torne-se aparente. Estas questões incluem a perda de estrutura, o rendimento significantemente reduzido, e semelhantes. Mesmo assim, a melhor solução é muitas vezes uma que resulta de um minucioso estudo do problema, ao invés de simplesmente ajustar as configurações para cima (aumento de buffer/comprimentos de fila, redução a latência de interrupção, etc.)

Para diagnosticar adequadamente um problema de desempenho de rede, use as seguintes ferramentas:

**netstat**

Um utilitário de linha de comando que imprime conexões de rede, tabelas de roteamento, estatísticas de interface, conexões mascaradas e associações de multicast. Ele recupera informações sobre subsistemas de rede a partir do sistema de arquivo **`/proc/net/`**. Estes arquivos incluem:

- **`/proc/net/dev`** (Informações de dispositivo)
- **`/proc/net/tcp`** (Informações de soquete TCP)
- **`/proc/net/unix`** (Informações de soquete de domínio Unix)

Para mais informações sobre **`netstat`**e seus arquivos de referncia de **`/proc/net/`**, consulte a página man **`netstat`**: **`man netstat`**.

**dropwatch**

Um utilitário de monitoramento que monitora pacotes despejados pelo kernel. Para mais informações, consulte a página man **`dropwatch`**: **`man dropwatch`**.

**ip**

Um utilitário para gerenciar e monitorar rotas, dispositivos, roteamento de política e túneis. Para mais informações consulte a página man **`ip`**: **`man ip`**.

**ethtool**

Um utilitário para exibir e modificar as configurações do NIC. Para mais informações, consulte a página man **`ethtool`**: **`man ethtool`**.

**/proc/net/snmp**

Um arquivo que exibe dados ASCII necessários para bases de informações de gerenciamento IP, ICMP, TCP e UDP para um agente **`snmp`**. Ele também exibe estatísticas de tempo real de UDP-lite.

O *SystemTap Beginners Guide* contém diversos scripts de amostra que você pode usar para o desempenho de rede do perfil e do monitor. Este guia está disponível a partir do link http://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/.

Após a coleta de dados relevantes sobre um problema de desempenho de rede, você deve ser capaz de formular uma teoria — e, felizmente, uma solução. [5] Por exemplo, um aumento de erros de entrada UDP em **/proc/net/snmp** indica que um ou mais soquetes que recebem filas estão cheias quando as pilhas de rede tentam enfileirar novas estruturas no soquete de um aplicativo.

Isso indica que os pacotes são afunilados *pelo menos* em uma fila de socket, o que significa que a fila do soquete drena pacotes muito lentamente, ou o volume de pacotes é muito grande para aquela fila de soquete. Se for o último caso, então verifique os logs de qualquer aplicação de rede intensiva, a procura de perda de dados - para resolver isso, você precisa para otimizar ou reconfigurar o aplicativo ofensivo.

### Soquete recebe tamanho de buffer

Soquete enviam e recebem tamanhos que são ajustados dinamicamente, de modo que raramente precisam ser editadas manualmente. Se uma análise mais aprofundada, como a análise apresentada no exemplo de rede SystemTap, **sk_stream_wait_memory.stp**, sugere que a taxa de dreno da fila de soquete é muito lenta, então você pode aumentar a profundidade da fila de soquete do aplicativo. Para fazer isso, aumente o tamanho de buffers de recepção por utilizados por soquetes, configurando um dos seguintes valores:

**rmem_default**

Um parâmetro de kernel que controla o tamanho do *default* de buffers de recepção usados pelo soquete. Para configurar isto, execute o seguinte comando:

```
sysctl -w net.core.rmem_default=N
```

Substitua o ***N*** pelo tamanho de buffer desejado, em bytes. Para determinar o valor para este parâmetro de kernel, visualize **/proc/sys/net/core/rmem_default**. Tenha em mente que o valor de **rmem_default** deve ser maior do que **rmem_max** (**/proc/sys/net/core/rmem_max**); se necessário, aumente o valor de **rmem_max**.

**SO_RCVBUF**

A opção de soquete que controla o tamanho *máximo* de buffers de recepção de soquete, em bytes. Para mais informações sobre **SO_RCVBUF**, consulte a página man para mais detalhes: **man 7 socket**.

Para configurar **SO_RCVBUF**, use o utilitário **setsockopt**. Você pode recuperar o valor atual **SO_RCVBUF** com o **getsockopt**. Para mais informações utilizando ambos utilitários, consulte a página man **setsockopt**: **man setsockopt**.

## 8.3. Visão Geral de Recepção de Pacotes

Para melhor analisar os funilamentos da rede e problemas de desempenho, você precisa entender como funciona a recepção de pacotes. Recepção de pacotes é importante para o ajuste de desempenho de rede, pois o caminho de recepção é onde as estruturas são muitas vezes perdidas. Estruturas perdidas no caminho de recepção pode causar uma penalidade significativa para o desempenho da rede.

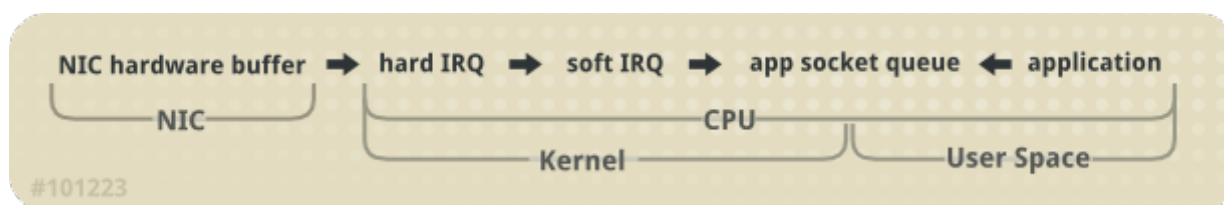



**Figura 8.1. Diagrama de caminho de recepção de rede**

O kernel do Linux recebe cada estrutura e sujeita-as a um processo de quatro passos:

1. *Recepção de Hardware*: a *placa de interface de rede* (NIC) recebe a estrutura a cabo. Dependendo de sua configuração do driver, o NIC tranferirá a estrutura para uma memória de buffer de hardware interno ou para um buffer de anel especificado.
2. *Hard IRQ*: o NIC declara a presença de uma estrutura de rede ao interromper a CPU. Isto faz com que o driver do NIC perceba a interrupção e agende a *operação IRQ leve*.
3. *Soft IRQ*: este estágio implementa o processo de estrutura de recepção e é executado no contexto de **`softirq`**. Isto significa que o estágio pré esvazia todos os aplicativos que estão executando em uma CPU específica, mas ainda permite que os IRQs rígidos a serem declarados.

   Neste contexto (em execução na mesma CPU como IRQ rígido, minimizando sobrecarga de bloqueio), o kernel realmente remove o quadro dos buffers de hardware NIC e o processa através da pilha de rede. A partir daí, o quadro é encaminhado, descartado ou passado para um soquete de escuta de alvo.

   Quando passado para um soquete, o quadro é anexado ao aplicativo que possui o soquete. Este processo é feito de forma iterativa até que o buffer de hardware NIC não tenha mais quadros, ou até que o *peso do dispositivo* ( **`dev_weight`** ). Para mais informações sobre o peso do dispositivo, consulte o Seção 8.4.1, “Buffer de Hardware NIC”
4. *Recepção de Aplicativo*: o aplicativo recebe o quadro e desinfileira-o de qualquer posse de soquete via chamadas POSIX padrão (**`read`**, **`recv`**, **`recvfrom`**). Neste ponto, os dados recebidos sob a rede não existem mais na pilha de rede.

### Afinidade de CPU/cache

Para manter o rendimento alto no caminho de recepção, é recomendado que você mantenha o cache L2 *hot*. Como descrito anteriormente, os buffers de rede são recebidos na mesma CPU como o IRQ que sinalizou a presença dos mesmos. Isto significa que os dados de buffer estarão no cache L2 daquela CPU que está recebendo.

Para aproveitar isto, coloque a afinidade do processo em aplicativos que deverão receber o maior número de dados sobre a NIC que compartilha o mesmo núcleo que o cache L2. Isto irá maximizar as chances de um acerto de cache, e, assim, melhorar o desempenho.

## 8.4. Resolvendo Filas Comuns/ Problemas de Perda de Quadro

O motivo mais comum para a perda de quadros é uma *saturação de fila*. O núcleo define um limite para o comprimento de uma fila, e, em alguns casos, a fila enche mais rapidamente do que drena. Quando isso ocorre por muito tempo, os quadros começam a ser descartados.

Como ilustrado em Figura 8.1, “Diagrama de caminho de recepção de rede”, existem duas grandes filas no caminho de recepção: o buffer de hardware NIC ea fila do socket. Ambas as filas precisam ser configurados adequadamente, para proteger contra saturação de filas.

### 8.4.1. Buffer de Hardware NIC

O NIC enche seu buffer de harware com quadros; o buffer é drenado pelo **softirq**, o qual declara o NIC via uma interrupção. Para interrogar o status desta fila, use o seguinte comando:

```
ethtool -S ethX
```

Substitua ***ethX*** pelo nome de dispositivo correspondente do NIC. Isto irá exibir quantos quadros foram despejados dentro do ***ethX***. Geralmente, um despejo ocorre porque a fila não possui mais espaço de buffer no qual armazena quadros.

Existem formas diferentes de resolver este problema, assim como:

**Tráfego de entrada**

Você pode ajudar a evitar saturação de fila diminuindo a velocidade do tráfego de entrada. Você pode fazer isto filtrando, reduzindo o número de grupos de multicast unidos, abaixando o tráfego de transferência e assim por diante.

**Comprimento da Fila**

Como forma alternativa você também pode aumentar o comprimento da fila. Isto envolve aumentar o número de buffers na fila específica para qualquer máximo que o driver permitir. Para fazer isto, edite os parâmetros de anel **rx**/**tx** de ***ethX*** usando:

```
ethtool --set-ring ethX
```

Anexe os valores de **rx** ou **tx** apropriados. Para mais informações, consulte **man ethtool**.

**Peso do dispositivo**

Também é possível aumentar a taxa na qual uma fila seja drenada. Para fazer isso, ajuste o *peso do dispositivo* do NIC adequadamente. Este atributo refere-se ao número máximo de quadros que a placa de rede pode receber antes do contexto **softirq** ter que render a CPU e se reagendar. É controlada pela variável **/proc/sys/net/core/dev_weight**.

A maioria dos administradores têm uma tendência a escolher a terceira opção. No entanto, tenha em mente que existem consequências para fazê-lo. Aumentar o número de quadros que podem ser recebidos a partir de um NIC em uma iteração implica ciclos extra de CPU, durante o qual nenhum aplicativo pode ser programado naquela CPU.

### 8.4.2. Fila de Soquete

Como a fila de hardware NIC, a fila de socket é preenchida pela pilha de rede a partir do contexto **softirq**. Os aplicativos então drenam as filas de seus soquetes correspondentes através de chamadas para **read**, **recvfrom**, e assim por diante.

Para monitorar o status desta fila, use o utilitário **netstat**; a coluna **Recv-Q** exibe o tamanho da fila. Geralmente falando, saturações em filas de soquete são gerenciadas da mesma forma que as saturações de buffer de hardware NIC (ex.: Seção 8.4.1, “Buffer de Hardware NIC”):

**Tráfego de entrada**

A primeira opção é diminuir o tráfego de entrada, configurando a taxa na qual se enche a fila. Para isso, você pode filtrar quadros ou despejá-los de forma pré-vazia. Você também pode

desacelerar o tráfego de entrada, diminuindo o peso do dispositivo [6].

**Profundidade da Fila**

Você também pode evitar saturação de fila de soquete aumentando a profundidade da fila. Para fazer isto, aumente o valor do parâmetro do kernel **rmem_default** ou a opção de soquete **SO_RCVBUF**. Para mais informações sobre ambos, consulte Seção 8.2, “Configurações de Rede Otimizadas”.

**Frequência de chamada de aplicativo**

Sempre que possível, otimize o aplicativo para realizar realizar chamadas com mais freqüência. Trata-se de modificar ou reconfigurar o aplicativo de rede para realizar chamadas POSIX mais freqüentes (como **recv**, **read**). Isso permite que um aplicativo drene a fila mais rápidamente.

Para muitos administradores, aumentar a profundidade da fila é a solução preferível. Esta é a solução mais fácil, mas nem sempre pode funcionar a longo prazo. A medida que as tecnologias de rede ficam mais rápidas, as filas de soquete continuam a encher mais rapidamente. Com o tempo, isso significa ter de voltar a ajustar a profundidade da fila de acordo.

A melhor solução é aumentar ou configurar o aplicativo para drenar dados do kernel mais rapidamente, mesmo que isso signifique enfileiramento dos dados no espaço de aplicação. Isso permite que os dados sejam armazenados de forma mais flexível, uma vez que pode ser trocado e paginado novamente quando necessário.

## 8.5. Considerações do Multicast

Quando várias aplicações escutam um grupo de multicast, o código do kernel que lida com quadros multicast é exigido pelo projeto para duplicar os dados da rede para cada soquete individual. Esta duplicação é demorada e ocorre no contexto **softirq**.

Adicionar vários ouvintes em um único grupo multicast, portanto, tem um impacto direto sobre o tempo de execução do contexto **softirq**. Adicionar um ouvinte a um grupo multicast implica que o kernel deve criar uma cópia adicional para cada quadro recebido para esse grupo.

O efeito disto é mínimo em baixo volume de tráfego e números ouvinte pequenos. No entanto, quando soquetes múltiplas escutam um grupo de multicast de alto tráfego, o aumento do tempo de execução do contexto **softirq** pode levar a despejos de quadros, tanto na placa de rede quanto na fila de socket. Aumento de tempos de execução do **softirq** traduzi em uma redução de oportunidade para que aplicações sejam executados em sistemas fortemente carregados, por isso a taxa em que os quadros multicast são perdidos aumenta à medida que o número de aplicações de escuta de grupos de multicast de alto volume aumenta.

Resolva esta perda de quadro otimizando suas filas de sockets e buffers de hardware NIC, como descrito em Seção 8.4.2, “Fila de Soquete” or Seção 8.4.1, “Buffer de Hardware NIC”. Como forma alternativa, você poderá otimizar o uso de um soquete de aplicativo. Para fazer isto, configure o aplicativo para controlar um soquete único e disseminar os dados de rede recebidos rapidamente para outros processo de espaço de usuário.

---

[4] Assegurar o cache affinity entre uma CPU e NIC significa configurá-los para compartilhar do mesmo cache L2. Para mais informações, consulte o Seção 8.3, “Visão Geral de Recepção de Pacotes”.

[5] Seção 8.3, “Visão Geral de Recepção de Pacotes” contém uma visão geral do pacote de viagem, o que deve ajudá-lo a localizar e mapear as áreas propensas a funilamento na pilha de rede.

[6] Peso do dispositivo do NIC é controlado via **/proc/sys/net/core/dev_weight**. Para mais informações sobre o peso do dispositivo e as implicações de ajustá-lo, consulte Seção 8.4.1, “Buffer de Hardware NIC”.

# Histórico de Revisões

| | | |
|---|---|---|
| **Revisão 4.0-22.2.400** | **2013-10-31** | **Rüdiger Landmann** |

Rebuild with publican 4.0.0

| | | |
|---|---|---|
| **Revisão 4.0-22.2** | **Wed Jun 26 2013** | **Glaucia Cintra** |

pt-BR translation completed

| | | |
|---|---|---|
| **Revisão 4.0-22.1** | **Thu Apr 18 2013** | **Chester Cheng** |

Tradução de arquivos sincronizados com a versão 4.0-22 de fontes do XML

| | | |
|---|---|---|
| **Revisão 4.0-22** | **Fri Feb 15 2013** | **Laura Bailey** |

Publicação para Red Hat Enterprise Linux 6.4.

| | | |
|---|---|---|
| **Revisão 4.0-19** | **Wed Jan 16 2013** | **Laura Bailey** |

Correções mínimas para obtenção de consistência (BZ#868404).

| | | |
|---|---|---|
| **Revisão 4.0-18** | **Tue Nov 27 2012** | **Laura Bailey** |

Publicação para Red Hat Enterprise Linux 6.4 Beta.

| | | |
|---|---|---|
| **Revisão 4.0-17** | **Mon Nov 19 2012** | **Laura Bailey** |

Foi adicionada a seção SME feedback re. numad (BZ#868404).

| | | |
|---|---|---|
| **Revisão 4.0-16** | **Thu Nov 08 2012** | **Laura Bailey** |

Foi adicionada a seção de rascunho em numad (BZ#868404).

| | | |
|---|---|---|
| **Revisão 4.0-15** | **Wed Oct 17 2012** | **Laura Bailey** |

Aplicando o feedback do SME para o descarte de bloco e foi movida a seção para uma subseção de Mount Options (BZ#852990).
Descrições de perfil de desempenho foram atualizadas (BZ#858220).

| | | |
|---|---|---|
| **Revisão 4.0-13** | **Wed Oct 17 2012** | **Laura Bailey** |

Descrições de perfil de desempenho foram atualizadas (BZ#858220).

| | | |
|---|---|---|
| **Revisão 4.0-12** | **Tue Oct 16 2012** | **Laura Bailey** |

Navegação de livro foi aprimorada (BZ#854082).
Foi corrigida a definição de ***file-max*** (BZ#854094).
Foi corrigida a definição de ***threads-max*** (BZ#856861).

| | | |
|---|---|---|
| **Revisão 4.0-9** | **Tue Oct 9 2012** | **Laura Bailey** |

Foram adicionadas recomendações do FSTRIM no capítulo de Sistemas de Arquivos (BZ#852990).
Descrição do parâmetro ***threads-max*** foi atualizada de acordo com o feedback do cliente (BZ#856861).
Uma nota sobre as melhorias de gerenciamento da fragmentação do GFS2 foi atualizada BZ#857782).

| | | |
|---|---|---|
| **Revisão 4.0-6** | **Thu Oct 4 2012** | **Laura Bailey** |

Foi adicionada uma nova seção no utilitário do numastat (BZ#853274).

| | | |
|---|---|---|
| **Revisão 4.0-3** | **Tue Sep 18 2012** | **Laura Bailey** |

Foi adicionada uma nota de capacidades do re. new perf (BZ#854082).

Foi corrigida a descrição do parâmetro file-max (BZ#854094).

| **Revisão 4.0-2** | **Mon Sep 10 2012** | **Laura Bailey** |
|---|---|---|

Foi adicionada uma seção do BTRFS e introdução básica ao sistema de arquivo (BZ#852978).
Integração do Valgrind foi anotada com o GDB (BZ#853279).

| **Revisão 3.0-15** | **Thursday March 22 2012** | **Laura Bailey** |
|---|---|---|

Foi adicionada e atualizada as descrições de perfis tuned-adm (BZ#803552).

| **Revisão 3.0-10** | **Friday March 02 2012** | **Laura Bailey** |
|---|---|---|

Foram atualizadas as descrições dos parâmetros threads-max e file-max (BZ#752825).
Foi atualizado o valor padrão do parâmetro slice_idle (BZ#785054).

| **Revisão 3.0-8** | **Thursday February 02 2012** | **Laura Bailey** |
|---|---|---|

Foi reestruturado e adicionado detalhes sobre o taskset e CPU de binding e alocação de memória com o numactl em Seção 4.1.2, “Ajustando Desempenho de CPU” (BZ#639784).
Foi corrigido o uso de links internos (BZ#786099).

| **Revisão 3.0-5** | **Tuesday January 17 2012** | **Laura Bailey** |
|---|---|---|

Correções pequenas em Seção 5.3, “Utilizando o Valgrind para o Uso de Memória de Perfil ” (BZ#639793).

| **Revisão 3.0-3** | **Wednesday January 11 2012** | **Laura Bailey** |
|---|---|---|

Consistência confirmada entre hiperlinks internos e externos (BZ#752796).
Adicionado Seção 5.3, “Utilizando o Valgrind para o Uso de Memória de Perfil ” (BZ#639793).
Adicionado Seção 4.1.2, “Ajustando Desempenho de CPU”e reestruturado Capítulo 4, *CPU* (BZ#639784).

| **Revisão 1.0-0** | **Friday December 02 2011** | **Laura Bailey** |
|---|---|---|

Lançamento para o GA do Red Hat Enterprise Linux 6.2.